Ellinger

COMMODORE



Das große Floppybuch

EIN DATA BECKER BUCH


Ellinger
COMMODORE
1571
&1570
Das große
Floppybuch
EIN DATA BECKER BUCH

ISBN 3-89011-124-6

Wichtiger Hinweis:
Die in diesem Buch wiedergegebenen Schaltungen, Verfahren und Programme werden ohne Rücksicht auf die Patentlage mitgeteilt. Sie sind ausschließlich für Amateur- und Lehrzwecke bestimmt und dürfen nicht gewerblich genutzt werden.

Alle Schaltungen, technischen Angaben und Programme in diesem Buch wurden von dem Autoren mit größter Sorgfalt erarbeitet bzw. zusammengestellt und unter Einschaltung wirksamer Kontrollmaßnahmen reproduziert. Trotzdem sind Fehler nicht ganz auszuschließen. DATA BECKER sieht sich deshalb gezwungen, darauf hinzuweisen, daß weder eine Garantie noch die juristische Verantwortung oder irgendeine Haftung für Folgen, die auf fehlerhafte Angaben zurückgehen, übernommen werden kann. Für die Mitteilung eventueller Fehler ist der Autor jederzeit dankbar.

Vorwort

Sehr geehrte Leserin, sehr geehrter Leser,

mit der C-1570/71 Floppy steht Ihnen eines der leistungsfähigsten 5¼-Zoll-Laufwerke, die es für Homecomputer gibt, zur Verfügung. Sie verarbeitet zwei unterschiedliche COMMODORE-Diskettenformate und eine Unzahl verschiedener CP/M Disketten. Ausserdem sind die COMMODORE-Floppies wohl die einzigen Laufwerke, die sozusagen einen eigenen Computer enthalten - nämlich eine selbständige Mikroprozessorsteuerung.

Das Floppy-Buch soll Ihnen helfen, alle Funktionen der C-1570 und der C-1571 kennenzulernen. Unter diesem Aspekt ist auch der Lesewegweiser auf der nächsten Seite entstanden. Denn mein Ziel ist es, Sie zu einem erfolgreichen Einsatz der Floppy hinzuführen. Ob Sie blutiger Anfänger oder gewiefter Profi sind, spielt dabei keine Rolle. Das 1570/71-Floppy-Buch ist allerdings nicht nur ein Lehrbuch, sondern vor allem auch ein Nachschlagewerk.
Das besondere Bonbon des Buches ist der Diskmonitor. Ein Programm, für das man normalerweise einige blaue Scheine hinlegen muß. Mit diesem sehr komfortablen Hilfmittel sind alle nur denkbaren Manipulationen auf der Diskette ausführbar. Überzeugen Sie sich doch einfach selbst. Kapitel 6.1.1 beschreibt alle Funktionen des Diskmonitors.
Auch die Profis werden am Floppy-Buch Ihre Freude haben. Vor allem das ROM-Listing stellt eine Weltpremiere dar. Noch nie wurde so ausführlich dokumentiert. Die Unterschiede zu bisherigen ROM-Listings sind nicht nur rein optisch feststellbar. Die Leistung des Listings zeigt sich erst im Detail. Einsprungpunkte und Aufrufadressen sind nur zwei Punkte, die in anderen Listings nicht zu finden sind (Näheres in 7.2).

Zum Schluß möchte ich Ihnen viel Spaß bei der Arbeit mit Ihrer C-1570/71 wünschen und dies umso mehr, als Ihnen mit diesem Buch ein erheblich tieferer Zugang zu den Leistungen dieser Floppy ermöglicht wird, als dies nur mit dem Handbuch je erreicht werden könnte.

Rainer Ellinger Oktober 1985

INHALTSVERZEICHNIS

Kapitel 1: Einführung für Einsteiger

Kapitel 3: Programmierung für Freaks

Kapitel 4: Die C-1571 und CP/M

Kapitel 5: Programmierung für Profis

Kapitel 8: Die Floppy im Griff - Übersichtstabellen

Lesehinweise - Ein Wegweiser durch das Buch

Sie haben nun ein sehr dickes Buch in der Hand, vollgestopft mit Informationen über die C-1570/71. Wie soll man sich da zurechtfinden?

Natürlich hat jedes Buch ein Inhaltsverzeichnis. Für spezielle Probleme gibt es außerdem ein Stichwortverzeichnis. Dies gehört zum Standard. Doch für ein wirklich hilfreiches Handbuch genügt das nicht. Deshalb ist zum Beispiel das Stichwortverzeichnis sehr umfangreich gestaltet worden und wird ergänzt durch eine thematische Zusammenfassung. Sie werden sich aber wohl trotzdem fragen, wo Sie mit der Lektüre dieses Buches beginnen sollen.

Aus diesem Grund gibt es die nebenstehende Lesetabelle - eine Art Wegweiser. Für jeden Typ von Leser, das heißt, jede Art von Vorkenntnissen wird ein Fahrplan zum Einstieg in das Buch angegeben. Dabei handelt es sich hauptsächlich um die Kapitel, die Ihren bisherigen Kenntnisstand erweitern. Sollten Sie in einem Kapitel einmal etwas nicht verstehen oder noch nie gehört haben, dann liegt das meistens daran, daß Sie eines der vorherigen Kapitel nicht gelesen haben. Durch das Buch zieht sich vom Anfang bis zum Ende ein roter Faden, da die Kapitel aufeinander aufbauen. Wer den Inhalt der vorherigen Kapitel nicht kennt, hat es unter Umständen schwer, den weiteren Teilen zu folgen.

Die einzelnen Spalten stehen für folgende Vorkenntnisse:

1 - Noch nie einen Computer gehabt, völliger Einsteiger. Noch nie mit Floppy gearbeitet.
2 - Umsteiger von anderen Computern. Aufsteiger vom C-64 ohne Floppy. BASIC gehört zum Repertoire.
3 - Umsteiger von anderem Computer mit Floppy.
4 - Umsteiger von VC-1541.
5 - Umsteiger von anderem Computer mit Floppy. Maschinensprache-Freak.
6 - Umsteiger von VC-1541. Maschinensprache-Freak

Lesewegweiser

1 -

Kapitel 1.1, 1.2, 1.3, 1.4, 1.5

2 -

Kapitel 1.1, 1.3, 1.4, 1.5, 5.1

3 -

Kapitel 1.2 und 1.3

4 -

Kapitel 2.1, 2.2, 2.3, 3.1 und 4

5 -

Kapitel 2.2, 3.1, 4, 5.2, 6, 7 und 8

6 -

Kapitel 6, 7 und 8

Alle anderen Kapitel sollten nach Interessenschwerpunkten genutzt werden. Ist das Basiswissen erst einmal vorhanden, bieten sie dem nun schon fortgeschrittenen Einsteiger problemlos weitere Informationen. Die Grundlagenkapitel zeigen sich als sinnvoll auch für den Profi, dem sie zum Nachschlagen und Überprüfen hilfreich sein werden.

KAPITEL 1
EINFÜHRUNG FÜR EINSTEIGER

1.1 DER ERSTE KONTAKT MIT DER FLOPPY

1.1.1 NACH DEM AUSPACKEN

Natürlich möchten Sie jetzt sofort loslegen und Ihre Floppy in Betrieb setzen. Trotzdem ist es ratsam, sich noch ein paar Momente zu gedulden und diese einleitenden Kapitel durchzulesen. Zuerst behandeln wir das Aufstellen und Anschließen der Floppy. In den folgenden Kapiteln geht es dann um das Datenmedium selbst - die Diskette. Sind Sie damit schon vertraut, so können Sie gleich mit Kapitel 1.1.3 fortfahren.

Zum Floppylaufwerk wird folgendes mitgeliefert:

Netzkabel
Verbindungskabel zum Computer
Test-/Demo-Diskette
Bedienungsanleitung

Schließen Sie die C-1570/71 zuerst mit dem 220-Volt-Netzkabel an die Stromversorgung an. Achten Sie darauf, daß das Gerät dabei ausgeschaltet ist. Danach müssen Sie das Floppylaufwerk mit dem Computer verbinden. Dazu dient das Kabel, das Sie vielleicht an Ihre Stereoanlage erinnert. Leider können aber die üblichen Hifi-Kabel nicht verwendet werden, da die Steckerbelegung anders ist.

Stecken Sie das Kabel zuerst in die entsprechende Buchse am Computer (siehe Abbildung 1).

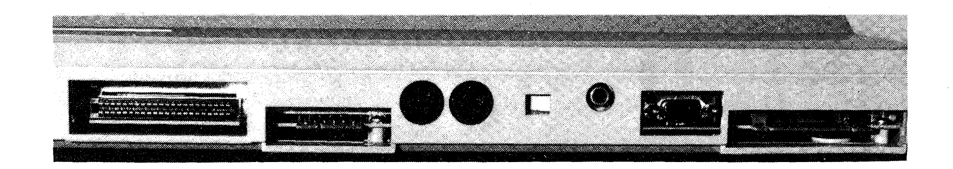


Abb. 1 Die Rückseite des C-128

Danach wird es einfach mit einer der beiden Buchsen an der Floppy verbunden. Jedes Gerät, das Sie an den Computer anschließen können (Floppy, Drucker, usw.) hat immer zwei Buchsen. Sonst könnten Sie nur ein Zusatzgerät am Computer betreiben, da dieser nur eine Ausgangsbuchse hat. Folglich dient eine der beiden Buchsen der Floppy als Eingang und die andere als Ausgang. Eine zweite Floppy oder ein Drucker werden dann an dieser Ausgangsbuchse der C-1570/71 angeschlossen. In welche der zwei Buchsen Sie das Signal vom Computer einspeisen, spielt keine Rolle. Wichtig ist nur, daß die andere Buchse dann nur als Ausgang verwendbar ist. Zwei Computer an einer Floppy - das funktioniert natürlich nicht.

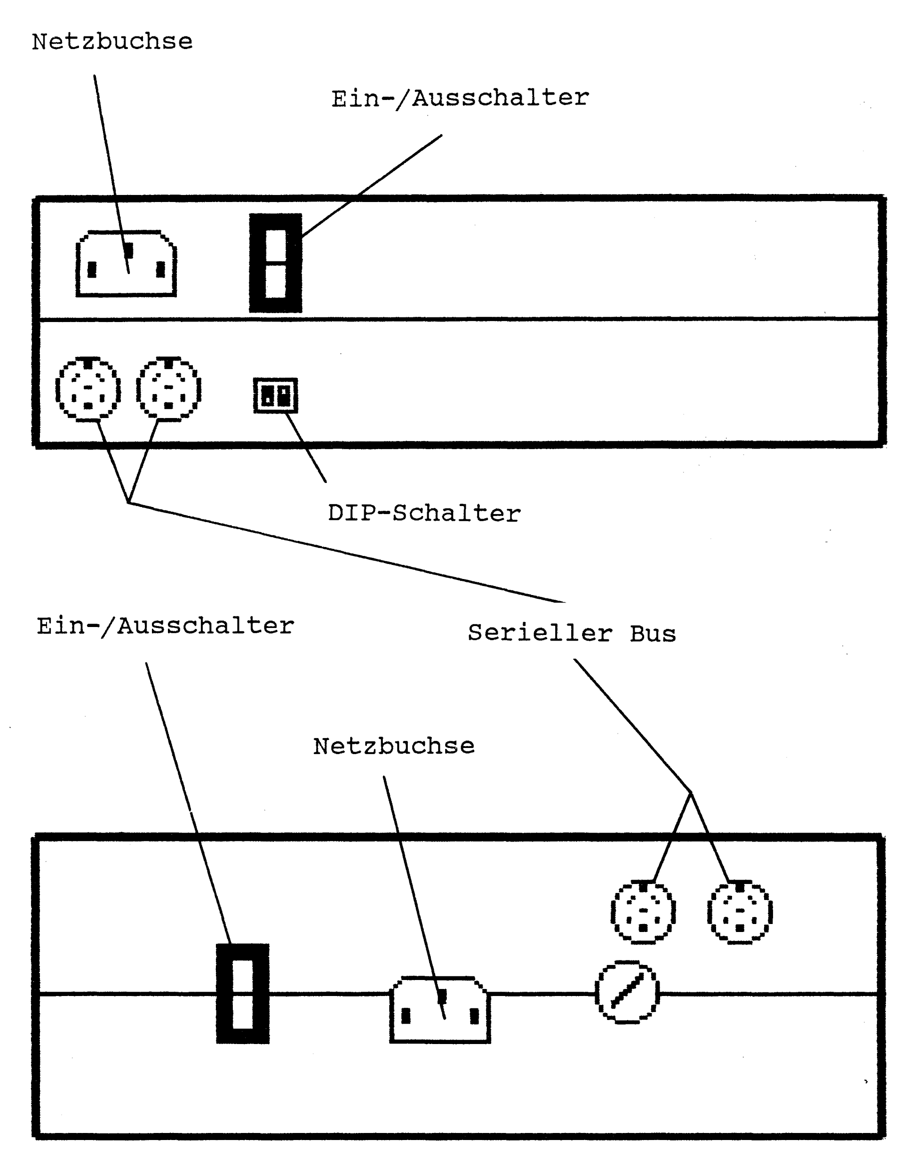


Abb. 2 Die Rückseite der Floppies 1570 und 1571

Wenn Sie eine 1571 besitzen, sollten Sie, bevor Sie das Gerät in Betrieb nehmen, einmal die beiden kleinen Schalter, auch DIP-Schalter genannt, an der Gehäuserückseite betrachten. Deren Funktion werden wir in 1.2.1 besprechen. Sie sollten beide in die

obere Schaltposition bringen. Bei der 1570 befinden sich diese Schalter im Gehäuseinneren und sind bereits richtig eingestellt.

Jetzt ist alles vorbereitet und Sie können das Laufwerk einschalten. Dabei leuchtet bei der C-1570 die grüne und bei der C-1571 die rote Betriebsanzeige auf und der Motor läuft kurz an. Die grüne (1570) bzw. rote (1571) Anzeige zeigt wie an Ihrem Rechner an, daß das Laufwerk eingeschaltet ist. Wenn Sie denn Einschaltvorgang genau beobachten, werden Sie bemerken, daß die andere Leuchtdiode kurz aufleuchtet. Ist dies alles eingetreten, dann funktioniert Ihre C-1570/71 einwandfrei. Blinkt die rote (1570) bzw. grüne (1571) Anzeige, dann hat die interne Selbsttestroutine einen Fehler gefunden. In diesem Fall hilft Ihnen das Fehlerkapitel 6.3.6 weiter.

Die rote (1570) bzw. grüne (1571) Leuchtdiode dient normalerweise auch als Betriebsanzeige. Sie zeigt an, daß auf die eingelegte Diskette gerade zugegriffen wird. Solange diese Anzeige leuchtet, dürfen Sie die Diskette nicht aus dem Laufwerk nehmen.

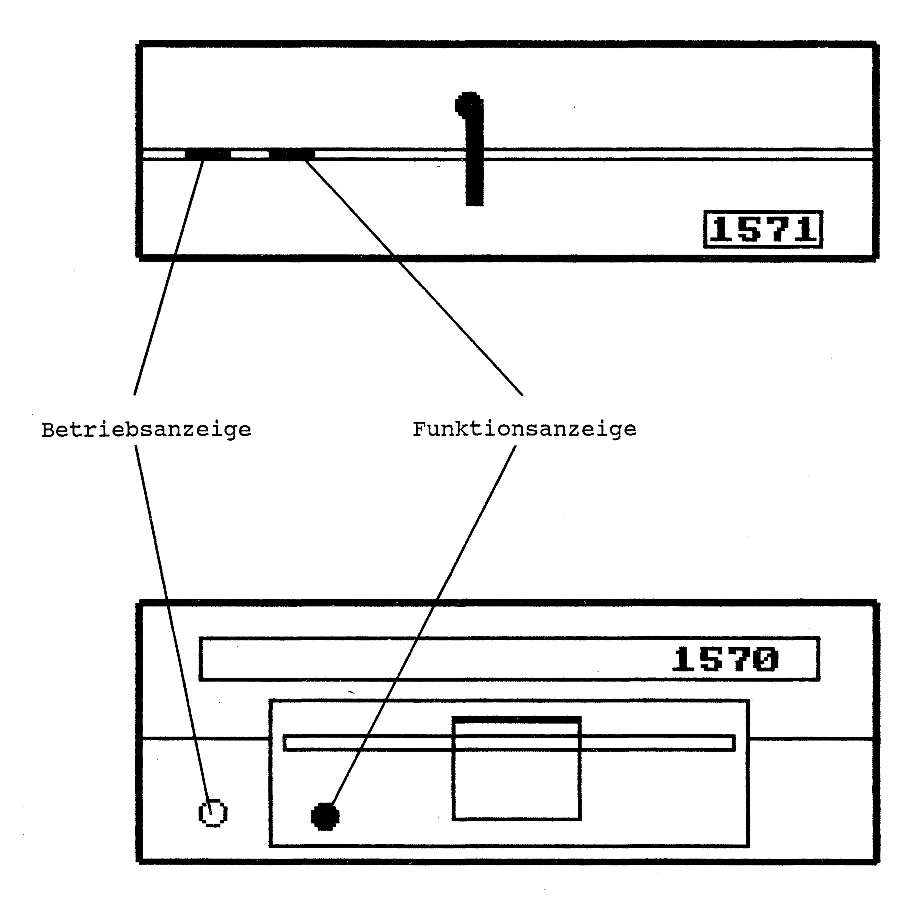


Abb. 3 Die Vorderseite der Floppies 1570 und 1571

1.1.2 WAS IST EINE DISKETTE?

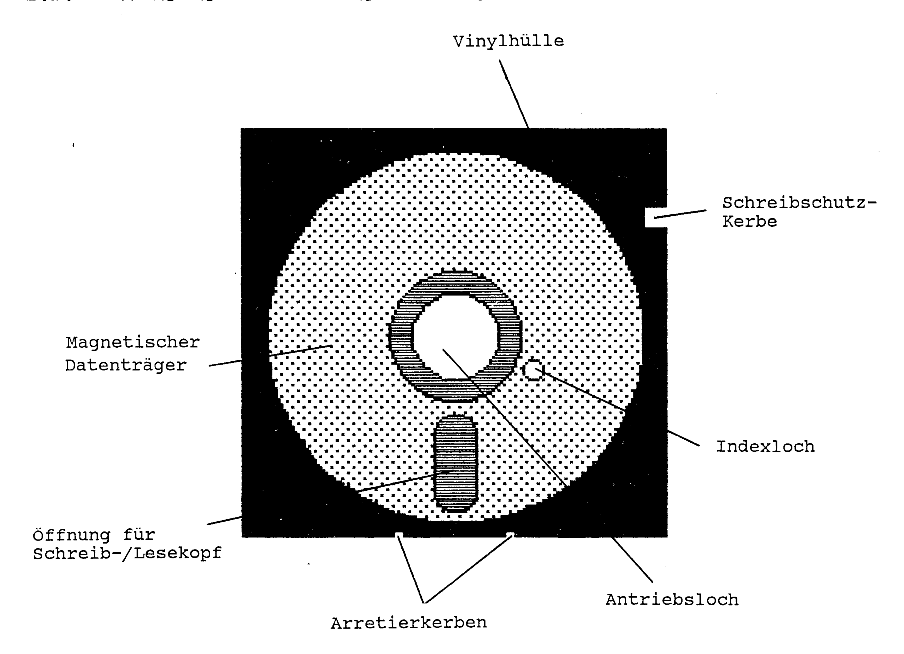


Abb. 4 Diskette

Abbildung 4 zeigt eine 5¼-Zoll-Floppy (zu deutsch etwa: 'Schlappscheibe'). Natürlich fällt sofort die große Öffnung an der unteren Seite auf. Hier kommt der eigentliche Datenträger, eine Magnetscheibe, zum Vorschein. Der Schreib-/Lesekopf im Laufwerk, der die Daten auf die Diskette transferiert und von ihr liest, ist so angebracht, daß er den Datenträger an dieser Stelle berühren kann. Die Disketten müßen mit dieser Schreiböffnung voran in das Laufwerk eingeschoben werden (Abbildung 5). Nun müßen Sie bei der C-1571 nur noch den Verschlußhebel in die senkrechte Position drehen. Die C-1570 hat einen Klappverschluß, der nach unten zu drücken ist, bis er einrastet. Die Diskette wird im Antriebsloch eingespannt. Dabei läuft der Motor kurz an. Dadurch soll die Diskette besser justiert werden, denn genau dies ist bei diesem Laufwerkstyp ein Problem. Achten Sie also darauf,

daß die Diskette bis zum Anschlag eingeschoben ist. Wenn Sie das Laufwerk in Betrieb nehmen, rotiert die Diskette mit etwa 5 Umdrehungen in der Sekunde (300 U/min). Daher daher befindet sich die Magnetscheibe in der Plastikhülle, die aber nicht nur dem mechanischen Schutz der empfindlichen Magnetschicht dient. Zusätzlich ist an der Innenseite ein Reinigungsvlies angebracht. Dieses entfernt Staubteilchen und kleine Verschmutzungen. Dabei müßen Sie bedenken, daß eine auf der Diskette gespeicherte Information gerade ein paar tausendstel Zentimeter groß ist - so wird auch schon die kleinste Verunreinigung zum Datenkiller. Behandeln Sie deshalb Ihre Disketten immer mit Sorgfalt und berühren Sie nie die eigentliche Magnetscheibe. Ihre Finger sind immer etwas fettig, und Fett kann das Reinigungsvlies nicht mehr entfernen. Beachten Sie auch die weiteren Hinweise auf der Rückseite jeder Papphülle, in die Sie Ihre Disketten nach Gebrauch immer wieder zurückstecken sollten. Achten Sie auch darauf, daß nie eine Diskette eingelegt ist, wenn Sie das Laufwerk ein- oder ausschalten (kleine unkontrollierte Spannungssprünge am Schreibkopf könnten wichtige Daten zerstören).

Der eckige Einschnitt an der rechten Seite ist die Schreibschutzkerbe (engl. Write Protect). Sie soll, wie der Name schon sagt, unbeabsichtigtes Schreiben oder Löschen von Daten verhindern. Dazu wird die Kerbe einfach zugeklebt. Doch auch der teuerste Klarsichtklebestreifen hilft nichts, da diese Kennung mit einer Lichtschranke abgefragt wird. Also genügen einfache kleine Klebeetiketten aus dem Schreibwarengeschäft oder die jeder Diskettenpackung beigelegten Klebeetiketten.

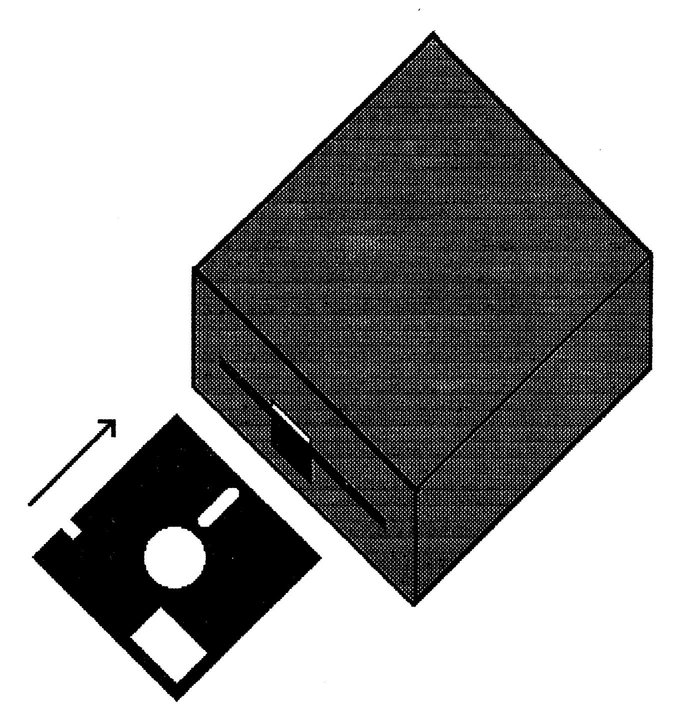


Abb. 5 So wird die Diskette in das Laufwerk eingelegt

Jetzt ergibt sich natürlich die Frage, wie Daten auf Diskette gespeichert werden? Dies geschieht selbstverständlich nicht willkürlich. Um die Daten auch wiederfinden zu können, wird die Diskette in Spuren beschrieben (Abbildung 6). Bringt man Daten auf die Diskette, so müßte man sich nur die Spurnummer merken und schon kann man die Daten auch wiederfinden. Eine Seite, einer im C-1570/71 Laufwerk verwendeten Diskette, besteht aus maximal 40 Spuren, mit je einer Kapazität von etwa 5000 Zeichen. Hat man aber zum Beispiel nur 1000 Zeichen, die man speichern möchte, so würde ein großer Teil der Spurkapazität verschenkt. Deshalb werden die Spuren nochmals unterteilt. Man zerlegt sie in verschiedene Sektoren, also Spurbereiche. Die ganze Diskette ist so in 40 Spuren unterteilt, die aus je 18 bis 21 Sektoren bestehen, wovon jeder wiederum 256 Zeichen aufnehmen kann. Diese Sektoren werden fortlaufend durchnumeriert. Wenn wir jetzt 1000 Zeichen speichern möchten, würden diese vier Sektoren der Spur belegen, wobei der letzte Sektor nicht ganz aus-

genützt würde. Die restlichen Sektoren der Spur würden weiterhin zur Datenspeicherung zur Verfügung stehen.

Wie aber erkennt das Floppylaufwerk, wo ein Sektor auf der Spur beginnt und wo er endet? Dies erfolgt über eine spezielle Markierung auf der Diskette - das Indexloch. Sie sicher schon das kleine Loch in der Diskette bemerkt, das direkt neben dem Antriebsloch liegt. Dieses kleine Loch wird mit einem Lichtstrahl im Laufwerk abgetastet.

Auf dem eigentlichen Datenträger befindet sich nun das oben genannte Indexloch. Gelangt das Indexloch durch das Drehen der Diskette in ihrer Hülle an die Stelle, an der das entsprechende Loch in der Hülle angebracht ist, so trifft der Lichtstrahl nicht mehr auf den Datenträger, sondern fällt durch das Indexloch hindurch auf eine Art Photozelle, wodurch der Schreib-/Lesekopf die Information erhält, daß sich genau unter ihm der erste Sektor jeder Spur befindet. Von dieser Stelle aus kann man problemlos die Position der anderen Sektoren berechnen. Dabei muß der Lesekopf nur die errechnete Zeit abwarten, um den Anfang eines beliebigen Sektors zu erreichen.

Doch welchen Vorteil hat diese Methode? Würde man vor jeden Sektor ein Indexloch einstanzen, dann wäre das doch viel einfacher? Der Nachteil liegt aber auf der Hand. Die Länge eines Sektors wäre unveränderbar vorbestimmt. Hat man nur ein Indexloch und zählt die Sektoren ab, so ist die Sektorgröße variabel. Man legt lediglich den Beginn des ersten Sektors fest. Die Position der weiteren Sektoren kann man aus der Sektornummer und der Sektorlänge errechnen.

Die Indexloch-Methode wird hauptsächlich im Betriebssystem CP/M verwendet. Disketten, die Sie im C-64- oder C-128-Modus verwenden, benötigen nicht einmal mehr das Indexloch. Damit das Laufwerk trotzdem weiß, wo ein Sektor beginnt, werden auf die Diskette spezielle Synchronisations-Marken (kurz: Sync-Markierung) geschrieben. Das Laufwerk erkennt diese Markierungen und kennt dadurch die Startstelle eines Sektors. Das Laufwerk weiß jetzt, wo ein Sektor beginnt. Doch wo ist der erste Sektor der Spur? Welche Nummer hat der Sektor, der gerade gefunden wurde?

Um auch dies zu erkennen, hat jeder Sektor einen sogenannten Header (zu deutsch etwa: Kopfstück). Der Header besteht einfach aus Zusatzinformationen, die vor den eigentlichen Daten abgespeichert werden. Das sind unter anderem die Nummer der Spur, auf der sich der Sektor befindet, und die Nummer des Sektors. Damit kann sich die Laufwerkselektronik auf der Diskette zurechtfinden. Soll ein bestimmter Sektor gelesen werden, so analysiert das Laufwerk einfach den nächsten Header. Nun kennt es die Spur, auf der der Kopf gerade steht und kann die richtige Spur ansteuern. Darauf muß nur noch, anhand der im Header abgelegten Nummern, der richtige Sektor gefunden werden.

Schon bald werden Sie sich dann fragen, auf welchen Sektoren Sie Ihre Daten abgelegt haben? Zu schnell hat man das vergessen. Und wenn man es sich notieren will, entsteht ein beträchtlicher Verwaltungsaufwand - eine C-128-Diskette hat immerhin über 1300 Sektoren. Diese Arbeit wird deshalb auch vom Laufwerk erledigt. Dazu werden auf der Diskette verschiedene Verzeichnisse angefertigt, zum Beispiel eine Tabelle, in der vermerkt ist, ob ein Sektor schon mit Daten belegt oder noch frei ist. Anhand dieser Tabelle beschreibt das Laufwerk dann die Sektoren.

Damit die Verwaltungsarbeit nicht zu zeitaufwendig, ist wird das Verzeichnis in einen speziellen Pufferspeicher im Laufwerk eingelesen und am Ende der Operation wieder zurückgeschrieben. Wechselt man die Diskette zwischenzeitlich, dann befindet sich das falsche Verzeichnis im Puffer und beim Speichern würden unter Umständen die falschen Sektoren beschrieben. Also muß ein Diskettenwechsel vom Laufwerk erkannt werden. Deshalb enthält der Header eines Sektors noch weitere Daten, so zum Beispiel zwei sogenannte Indentifikationszeichen (kurz: ID). Auf jeder Diskette steht eine andere Zeichenkombination. Das Laufwerk merkt sich immer die Zeichen des letzten gelesenen Headers. Wird der nächste Sektor gelesen, so wird die ID dieses Headers mit der letzten verglichen. Hat sich die ID geändert, dann weiß das Laufwerk, daß die Diskette gewechselt wurde.

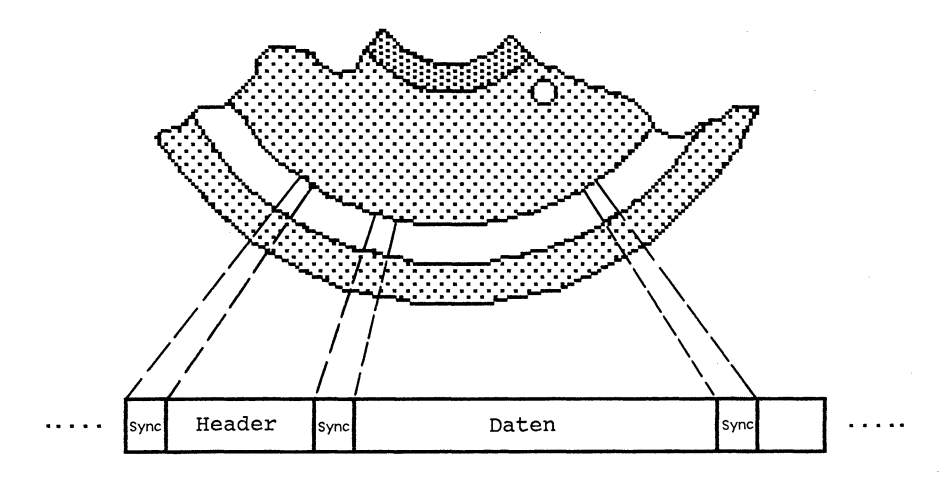


Abb. 6 Der Aufbau einer Diskettenspur

1.1.3 DIE DISKETTENFORMATE

Wie Sie gesehen haben, gibt es viele Möglichkeiten, wie eine Diskette aufgebaut sein kann: Indexloch- oder Sync-orientiert; mit 128, 256, 512 oder 1024 Bytes (Zeichen) pro Sektor; mit verschiedenen Sektorzahlen pro Spur usw. Darüberhinaus gibt es noch verschiedene Laufwerkstechniken. Die C-1570/71 beschreibt 40 Spuren pro Diskettenseite. Es gibt aber auch Laufwerke, die auf einer Seite 80 Spuren beschreiben können (höhere Spurdichte). Außerdem gibt es unterschiedliche Aufzeichnungsverfahren. Dies sind hauptsächlich verschiedene Datenraten, die deshalb als Single Density (einfache Schreibdichte) oder Double Density (doppelte Dichte) bezeichnet werden. Schließlich gibt es noch die Verzeichisse über Sektor- und Diskettenbelegung. Deren Aufbau hängt völlig vom verwendeten Computertyp ab, letztendlich also vom Hersteller des Computers.

Diese Vielfalt an Variationsmöglichkeiten hat zur Folge, daß es mehr Diskettenformate als Computerproduzenten gibt. Das heißt, daß Sie C-1570/71 Disketten, die im C-128-Modus verwendet

werden, nicht einfach im C-64-Modus verwenden können - ganz zu schweigen von anderen Computertypen.

Wo bekommt man nun Disketten mit diesem speziellen COMMODORE-Format? Habe ich schon die falschen gekauft?

Nein, Disketten im COMMODORE-Format kann man nicht kaufen. Worauf Sie lediglich achten müssen, ist die Zahl der Spuren, die die Disketten aufnehmen können. Da die C-1570/71 40 Spuren verwendet und dies die kleinst mögliche Anzahl ist, können Sie da gar nichts falsch machen. Disketten für 80 Spuren sind nicht erforderlich und kosten mehr (da höherwertiger). Das einzige, was Sie unbedingt beachten müssen ist, daß die Disketten für doppelte Schreibdichte geeignet sind (Double Density; kurz: DD), die einfache Schreibdichte reicht nämlich nicht aus. Ausserdem müssen sie beidseitig verwendbar sein (Double Sided; kurz: DS). Disketten, die Sie kaufen, sind immer leer. Wie kommen dann aber das Commodore-Format, die Sync-Markierungen, usw. auf die Diskette? Diese Arbeit erledigt die Floppystation und wird als Formatierung bezeichnet. Jede Diskette müssen Sie also, bevor Sie sie verwenden können formatieren. Mehr darüber in Kapitel 1.2.2.

DATEN DER COMMODORE-FORMATE

Format	1541/1570	1571
verwendete Diskettenseiten	1	2
max. Bits pro Sekunde	307692	307692
Gesamtzahl der Sektoren	683	1366
Zahl der freien Sektoren	664	1328
Zeichen pro Sektor	256	256
Zahl der Sektoren pro Spur		
Spur 1 - 17	21	21
Spur 18 - 24	19	19
Spur 25 - 30	18	18
Spur 31 - 35	17	17

DATEN DER CP/M-FORMATE

Diskettenseiten	1	2
Bits pro Sekunde	500000	500000
Zahl der Sektoren pro Spur		
128 Bytes pro Sektor	26	26
256 Bytes pro Sektor	16	16
512 Bytes pro Sektor	9	9
1024 Bytes pro Sektor	5	5
Gesamzahl der Sektoren		
128 Bytes pro Sektor	1040	2080
256 Bytes pro Sektor	640	1280
512 Bytes pro Sektor	360	720
1024 Bytes pro Sektor	200	400

1.2 DIE FLOPPY UND COMMODORE BASIC

1.2.1 VON BASIC 2.0 BIS BASIC 7.0

Um die C-1570/71 nun zum Arbeiten zu bringen muß man ihr Befehle erteilen. Dies ist nicht sehr kompliziert. Geben Sie einfach den Befehl in den Computer ein und betätigen Sie die RETURN-Taste. Durch diese Taste weiß der Computer, daß der Befehl ausgeführt werden soll. Wie Sie sicher schon wissen, heißt die Sprache, in der Ihr C-128 die Befehle erhalten möchte, BASIC.

Doch mit der Programmiersprache BASIC ist es wie mit dem Deutschen. Jede spricht zwar Deutsch, trotzdem haben es ein Norddeutscher und ein Bayer schwer, sich miteinander zu verständigen. Genauso gibt es auch bei BASIC eine Vielzahl von Dialekten. Die Grundbefehle sind zwar meistens gleich, doch die etwas anspruchsvolleren Kommandos sind bei jeder Version etwas anders. Selbst die BASIC-Versionen eines einzelnen Computerherstellers, in unserem Fall COMMODORE, verstehen sich auf dem Gebiet der Floppy-Befehle nicht miteinander. Die folgende Tabelle zeigt die verschiedenen COMMODORE BASIC-Versionen und die dazugehörigen Computer (nach Erscheinen geordnet):

Computer	Version
PET 2000	BASIC 1.0
CBM 3000	BASIC 3.0
CBM 8000	BASIC 4.0
VC-20 / C-64	BASIC 2.0
C-16 / Plus 4	BASIC 3.5
C-128	BASIC 7.0

Die Versionsnummern geben nicht etwa die Erscheinungsfolge an. Vielmehr ist diese Zahl ein Maß für das Niveau des BASIC. So ist BASIC 4.0 eben etwas leistungsfähiger als BASIC 2.0. Doch auch hier gibt es keine Regel ohne Ausnahme. BASIC 3.5 müßte eigentlich 4.5 heißen, weil es noch wesentlich mehr leistet als das

BASIC der CBM 8000er. Wie Sie auch sehen, ist die Zahl 7.0 weit von den übrigen Versionsnummern entfernt, genauso weit, wie das Niveau des C-128-BASIC - das leistungsfähigste, das COMMODORE je herausgebracht hat.

Für uns spielt die Version 3.0 eine entscheidende Rolle. Hier besteht nämlich ein wichtiger Einschnitt. Alle BASIC-Versionen größer als 3.0 haben sehr komfortable Floppy-Befehle. Bei den anderen Versionen ist die Floppybedienung etwas umständlicher. Die Syntax der Versionen, die kleiner gleich 3.0 sind (in diesem Buch immer BASIC < 3.0 genannt), wird natürlich auch von den großen Versionen verstanden. Die zusätzlichen Floppy-Befehle bei BASIC > 3.0 funktionieren aber nicht auf den Rechnern mit kleineren Versionen. In den folgenden Kapiteln werden immer beide Befehle, sowohl von BASIC < 3.0, als auch von BASIC > 3.0, angegeben.

Zu guter letzt gibt es noch eine dritte Möglichkeit, Floppyfunktionen zu verwenden - im eingebauten Maschinensprache-Monitor. Die Syntax dieser Befehle ist ähnlich der BASIC-Versionen kleiner 3.0 und zusätzlich angegeben.

Alles scheint kein Problem zu sein. Man gibt einen Befehl ein und schon beginnt die C-1570/71 mit der Arbeit. Doch, was passiert, wenn zwei Laufwerke an den Computer angeschlossen sind? Woher weiß der Computer, für welches Laufwerk der Befehl gelten soll? Aus diesem Grund erhält jedes an den C-128 angeschlossene Gerät eine Nummer. Die Floppy hat dabei normalerweise die Nummer 8, ein Drucker hat die Nummer 4 und der Kassettenrekorder Nummer 1. Diese Nummer nennt man Geräteadresse. Wenn Sie nun ein zweites 1570/71-Laufwerk haben kann dieses natürlich nicht auch die Adresse 8 erhalten. Man ihm eine andere Adresse zuordnen. Dazu diehnen die zwei kleinen DIP-Schalter. Diese befinden sich bei der C-1571 unter dem Ein-/Ausschalter an der Rückseite der Floppy. Mit einem Bleistift können Sie sie problemlos verstellen. Bei der C-1570 ist dies etwas komplizierter. Die Schalter befinden sich auf der Platine im Gehäuseinneren, an der rechten Seite. Um sie zu verstellen, müssen Sie also das Gehäuse aufschrauben. Dabei ist zu beachten, daß der Hersteller eventuell Garantieleistungen verweigert, da sie

die Floppy eigenmächtig geöffnet haben (obwohl man das kaum feststellen kann).

Die einzelnen Schalterstellungen entsprechen folgenden Geräteadressen:

Schalter 1 links	Schalter 2 rechts	Geräte-adresse
oben	oben	8
unten	oben	9
oben	unten	10
unten	unten	11

Die DIP-Schalter werden immer nach dem Einschalten (Reset) abgefragt. Sie müssen also die Floppy einmal Aus- und wieder Einschalten, damit sie die neue Adresse annimmt.

Veranschaulichen wir uns jetzt einmal den Datenverkehr zwischen Computer und Floppy. Die C-1570/71 kann ja nicht nur Programme speichern, sondern auch Dateien verwalten. Nehmen wir an, es würde gerade mit zwei Dateien gleichzeitig gearbeitet und man möchte in eine der Dateien neue Daten schreiben. Wenn man die Daten nun übermittelt, woher weiß die Floppy, in welche Datei sie gehören? Oder vielleicht gehören sie ja auch zu einem Programm, daß wir zwischenzeitlich abspeichern wollen?

Um dieses Problem zu lösen, gibt es zur Floppy sogenannte Datenkanäle. Jeder der Kanäle wird dabei nur für bestimmte Aufgaben benutzt. Man kann das etwa mit den Funkfrequenzen vergleichen. Auf der einen Frequenz gibt es nur Polizeifunk und der andere Kanal ist nur für die Amateurfunker.
Bei den COMMODORE-Floppies gibt es insgesamt 16 Kanäle. Davon können aber überlicherweise nur drei bis vier gleichzeitig benutzt werden. Den Kanälen wird, wie den einzelnen Geräten, eine Nummer zugeteilt. Die folgende Tabelle zeigt die Verwendung der Kanäle.

Kanalnummer	Funktion
0	Load
1	Save
2 - 14	für Dateien
15	Befehlskanal

Um einen bestimmten Kanal zu aktivieren, gibt es beim C-64 wie auch beim C-128 den 'Open'-Befehl. Die Syntax des Befehls sieht folgendermaßen aus:

OPEN X,Y,Z,"daten/name"

Die Parameter Y und Z kennen Sie schon. Y ist die Geräteadresse der angesprochenen Floppy, die man auch Primäradresse nennt. Z gibt die Kanalnummer an, die Sekundäradresse. Danach kann man schon beim Eröffnen Daten oder den Namen einer Datei übermitteln. Soll dies unterbleiben, lautet der Befehl OPEN X,Y,Z. Solche Angaben, die nicht unbedingt zur Ausführung eines Befehls erforderlich sind, nennt man 'optional'. Der Begriff 'optionaler Parameter' taucht auch in den folgenden Kapiteln noch oft auf. Derartige Informationen können weggelassen werden, wenn man sie nicht benötigt. Das Komma, das die Angabe von den übrigen Parametern abgrenzt, muß natürlich auch entfallen.

Was Sie noch nicht kennengelernt haben, ist der Parameter X. Dies ist eine beliebige Zahl zwischen 0 und 255. Diese Zahl wird dem Kanal zugeordnet. Wollen wir wieder Daten an den Kanal senden, dann geschieht das mit dem PRINT#-Befehl. Jetzt genügt die Angabe der logischen Kanalnummer (X), und der Computer kennt die zugeordneten Primär- und Sekundäradressen. Ist die Arbeit auf einem Kanal abgeschlossen, dann muß dies dem Computer mit 'CLOSE X' mitgeteilt werden. Die logische Kanalnummer verkürzt also die Angaben bei jedem weiteren Befehl, wodurch die Arbeit mit der Floppy einfacher wird.

Diese Kanalbefehle sind besonders bei der Dateiverwaltung wichtig. Daher werden sie in 1.4 und 1.5 ausführlich besprochen.

1.2.2 HEADER - FORMATIEREN EINER DISKETTE

BASIC > 3.0:	HEADER "diskettenname",Dx,Iyy,Uz
Abkürzung:	heA
BASIC < 3.0:	OPEN 1,z,15,"Nx:diskname,yy"
Monitor:	@z,Nx:diskname
Parameter (optional):	
Dx : x	= Laufwerksnummer (0/1)
Iyy : yy	= zwei ID-Zeichen
Uz : z	= Geräteadresse des Laufwerks (4..15)

In Kapitel 1.1.3 sprachen wir vom Formatieren. Jede neue, leere Diskette muß formatiert werden, bevor man sie zur Datenspeicherung verwenden kann. Dabei werden die Sync-Markierungen, Header und Sektoren angelegt.

Sollen neue Disketten formatiert werden, muß eine ID angegeben werden. Diese beiden Identifikationszeichen erlauben es dem Laufwerk, Disketten zu unterscheiden und festzustellen, wenn eine neue Diskette eingelegt wird. Deshalb ist es wichtig, bei jeder Diskette eine andere Zeichenkombination zu verwenden. Die ID-Informationen werden beim Formatieren in jedem Sektorheader abgelegt. Weiter sind die ID-Zeichen auch im Inhaltsverzeichnis (Titelzeile) der Diskette angegeben. Soll die ID einmal nachträglich geändert werden, dann hilft der in 6.1 beschriebene Diskmonitor weiter.

Weiter müssen Sie beachten, daß in BASIC 7.0 nicht alle Zeichenkombinationen richtig funktionieren. Dies liegt daran, daß der Computer die Zeichen als BASIC-Befehl interpretiert und anstelle der Zeichen die entspechenden Kurz-Codes verwendet.

Keine Angst, es gibt schon noch genügend Kombinationen, die erlaubt sind. Zusammen mit den Ziffern 0-9 sind es 1296 Möglichkeiten. Wenn Sie davon 100 nicht benützen können, dürfte Sie das nicht allzu sehr einschränken. Ausserdem kann man auch auf den BASIC-<3.0-Befehl ausweichen.

Folgende Kombinationen funktionieren nicht:
(Bitte Klein- und Großschreibung beachten)

```
on fn to aP aU bA bE bL bO bS bU cA cI cO dC dL dO dR dS dV eN fA fE fI fR
gR gS hE jO kE mO pA pE pL pO pU rC rD rE rR rS rU rW sC sL sO sP sS sT sW
tE tR vO wI xO aB aN aS aT cH cL cM cO dA dE dI eN eX fO fR gE gO iN lE lI
lO mI nE nO oP pE pO pR rE rI rN rU sA sG sI sP sQ sT sY tA tH uS vA vE wA
```

Wenn Sie den Header-Befehl ohne ID aufrufen, wird die Diskette nicht neu formatiert, sondern nur gelöscht. Dabei gehen trotzdem, wie beim vollständigen Formatieren, alle Daten verloren. Deshalb fragt der Computer vorher mit 'Are you sure ?', ob Sie sicher sind, daß die Diskette formatiert oder gelöscht werden soll. Wenn ja, beantworten Sie die Frage mit 'y' wie 'Yes' (ja). Es kann natürlich auch sein, daß sie den Header-Befehl im Programm verwenden. Dann wird allerdings nicht gefragt, ob Sie auch wirklich sicher sind, die Diskette zu formatieren. Der Befehl wird sofort ausgeführt. Für Sicherheitsabfragen müssen Sie in Ihrem Programm selbst sorgen.

Wie Sie bereits aus 1.1.3 wissen, sind die Diskettenformate im C-64- und im C-128-Modus nicht identisch. CP/M-Disketten haben wiederum einen völlig anderen Aufbau. Die Abweichungen im 64er und 128er Modus beruhen darauf, daß sich das Floppylaufwerk im 64er Modus oder an einem C-64 angeschlossen wie eine 1541-Floppy verhält. Befindet sich der Computer im 128er Modus, so schaltet sich das Laufwerk in den 1571-Modus. Der größte Unterschied zwischen den beiden Modi ist die Diskettenkapazität. Bei der 1571 werden Disketten beidseitig beschrieben, während die 1541 immer nur eine Seite verwendet, da sie nur einen Schreib-/Lesekopf hat. Trotzdem können 1571-Disketten auch im C-64-Modus verwendet werden - vorausgesetzt es ist ein C-1571-Laufwerk angeschlossen. Die C-1570 wiederum kennt keine 2. Seite und verhält sich immer wie eine 1541.

1.2.3 DLOAD/RUN - BASIC-PROGRAMME LADEN UND AUSFÜHREN

BASIC > 3.0:	DLOAD "programmname",Dx,Uy RUN "programmname",Dx,Uy
Abkürzung:	dL / rU
BASIC < 3.0:	LOAD "x:programmname",y RUN "programm" ist nicht möglich
optionale Parameter: x	
Monitor:	L"x:programmname",yy,aaaa autom. Starten nicht möglich
aaaa = Anfangsadresse des Programms	
Parameter (optional):	
Dx : x	= Laufwerk (0/1)
Uy : y	= Geräteadresse (4..15)

Jetzt wird es ernst. Nun geht es um die ersten Befehle zum Arbeiten mit der Diskette. Dabei dürfte das Laufwerk wohl hauptsächlich zum Speichern von Programmen benutzt werden.

Deshalb wollen wir zuerst den Befehl besprechen, mit dem man ein Programm von der Diskette in den Computer einliest. Dieser heißt in seiner einfachsten Form

```
DLOAD "programmname"
```

Von dem englischen Ausdruck 'load' abgeleitet spricht man im Computerjargon meistens vom 'Programm laden'. Das 'd' bei 'dload' steht für 'Disk/Diskette'. Es handelt sich bei Dload nur um

eine Spezialversion des normalen Load-Befehls, bei der man die lästige Geräteadresse nicht angeben muß.

Befindet sich gewünschte Programm auf der Diskette, dann wird in den Speicher des Computers geladen. Wurde das Programm nicht gefunden, so reklamiert der Computer mit

FILE NOT FOUND

Dies geschieht auch, wenn der Dload-Befehl in einem Programm verwendet wird. Ausserdem wird das Programm unterbrochen und der Computer kehrt in den Direktmodus zurück.

Sie können diesen Befehl einmal mit Ihrer Test-/Demo-Diskette ausprobieren. Versuchen Sie, die verschiedenen Programme von der Diskette zu laden. Soll das Programm sofort nach dem Laden gestartet werden, dann ist anstelle von 'Dload' einfach 'Run' zu verwenden.

Eine noch größere Bequemlichkeit ermeglichen die Tasten 'SHIFT + RUN/STOP'. Betätigen Sie beide gleichzeitig, so erscheint auf dem Bildschirm automatisch der Befehl 'Dload ":*" (in abgekürzter Form) und danach 'Run'. Dadurch liest der Computer das erste Programm auf der Diskette und startet es.

Natürlich gibt es bei den Ladebefehlen zwischen dem 64er und dem 128er Modus Unterschiede. Der gravierendste Gegensatz besteht in der Übertragungsgeschwindigkeit. Trotten beim C-64-Modus die Zeichen mit der gemächlichen Geschwindigkeit von 400 Zeichen pro Sekunde über den Bus, so wird der C-128-Bus mit einer Übertragungsrate von 3500 Bytes pro Sekunde zur Rennbahn. In der Praxis bedeutet dies, daß zum Beispiel ein Grafikbild nicht mehr in etwa 20 Sekunden geladen wird, sondern schon nach 3 Sekunden zur Verfügung steht.

Außerdem verhalten sich die Ladebefehle auch beim Nachladen (Overlay) von Programmen unterschiedlich. Während der C-64 dabei normalerweise alle Variablen 'vergißt', bleiben diese beim C-128 alle erhalten. So kann man große Programme problemlos in mehrere Teile zerlegen.

1.2.4 DSAVE - BASIC-PROGRAMME SPEICHERN

BASIC > 3.0:	DSAVE "programmname",Dx,Uy
Abkürzung:	dS
BASIC < 3.0: optionale Parameter: x	SAVE "x:programmname",y
Monitor: aaaa/bbbb = Anfangs- und Endadresse des Programms	S"x:programmname",yy,aaaa,bbbb+1
Parameter (optional): Dx : x = Laufwerk (0/1) Uy : y = Geräteadresse (4..15)	

Soll ein Programm vom Computerspeicher auf die Diskette übertragen werden, dann geschieht das wie bei 'dload' - hier aber mit dem Befehl 'dsave'. Wenn zum Beispiel ein BASIC-Programm abgespeichert werden soll, muß ein passender Name für das Programm gefunden werden. Nehmen wir an, es soll 'Minitest' heißen. Der Speicherbefehl lautet:

```
DSAVE "minitest"
```

Dabei muß beachtet werden, daß der Name nicht mehr als 16 Zeichen lang sein darf und bisher kein gleichnamiges Programm auf der Diskette existiert. Ausserdem gibt es einige Zeichen, die in Programm- und Dateinamen nicht verwendet werden dürfen. Diese Zeichen gehöhren meistens zu Befehlen oder haben Steuerfunktionen. Bei deren Gebrauch kann das Programm später nicht mehr geladen werden, weil die Floppy den Namen als Befehl interpretiert. Es handelt sich um die Zeichen:

, : ? * # & @

Bald wird es allerdings vorkommen, daß Sie Ihr Programm etwas geändert haben und die neue Version unter dem gleichen Namen abspeichern möchten. Dazu gibt man als erstes Zeichen '@' (at-Zeichen) an, danach den gesamten Namen des Programms, das man überschreiben will, also zum Beispiel

DSAVE "@minitest"

Bei dieser Spezialfunktion wird zuerst das neue Programm abgespeichert und danach die alte Version gelöscht. Deshalb muß auf der Diskette immer noch soviel Platz sein, daß man das Programm zusätzlich (nicht ersetzend) abspeichern kann. Leider gibt es bei dieser Sache auch einen Wermutstropfen. Ist die Diskette fast voll, so arbeitet diese Ersetzungsfunktion fehlerhaft und es kann passieren, daß Ihr Programm verlorengeht. Sie sollten den '@' also mit Bedacht einsetzen - oder lieber gar nicht. In BASIC <3.0 sowie beim Monitor muß nach dem '@' ein Doppelpunkt folgen, um den Programmnamen abzutrennen (z.B save "@:Minitest").

Die Speicherzeiten der 1570/71 sind allerdings nicht so berauschend, wie die Ladegeschwindigkeit. Dies verläuft genauso langsam wie bei der 1541. Ausserdem ist das Speichern generell langsamer als das Laden, da nach jedem Schreiben nochmals geprüft wird, ob die Daten auf der Diskette richtig gespeichert sind.

Speicher und Ladezeiten im Vergleich

	Lesen	Schreiben
C-64 10KByte-Programm	0:27	0:30
C-64 10KByte-Datei	2:35	2:45
C-128 10KByte-Programm	0:03	0:25
C-128 10KByte-Datei	3:05	2:50

1.2.5 DVERIFY - PROGRAMM ÜBERPRÜFEN

BASIC > 3.0:	DVERIFY "programmname",Dx,Uy,z
Abkürzung:	dV
BASIC < 3.0:	VERIFY "x:programmname",y,z
optionale Parameter: x	
Monitor:	V"programmname",yy,aaaa
aaaa = Anfangsadresse des Programms	
Parameter (optional):	
Dx : x	= Laufwerk (0/1)
Dy : y	= Geräteadresse (4..15)
z : z	= 0: relativ laden 1: absolut laden

Dieser Befehl dient der Überprüfung eines Programms auf der Diskette. Er vergleicht es mit dem, das sich im Computer befindet. Stimmen beide überein, so meldet der Computer 'Ok'. Wenn nicht, erfolgt die Anzeige eines 'Verify Error'.

'Dverify' stammt eigentlich noch aus der Kassettenanwendung. Aufgrund der mangelhaften Speichersicherheit war es immer ratsam, das abgespeicherte Programm nochmals zu überprüfen. Im Zeitalter der erschwinglichen Diskettenstationen ist diese Funktion eigentlich überflüssig.

Außerdem überprüft die Floppy, nachdem sie Daten in einen Sektor geschrieben hat, diesen nochmals auf Richtigkeit. Verify wird von der 1570/71-Floppy bei jedem Speichervorgang automatisch durchgeführt. Deshalb benötigt das Speichern von Programmen auch etwas mehr Zeit als das Laden.

1.2.6 BLOAD/BSAVE - MASCHINENPROGRAMME SPEICHERN/LADEN

BASIC > 3.0:	BLOAD "programmname",Dx,Uy,ON Bz,Pa BSAVE "programmname",Dx,Uy,ON Bz,Pa TO Pb
Abkürzung: bL / bS	
BASIC < 3.0:	LOAD "x:programmname",y,1 SAVE nicht vorhanden
optionale Parameter: x	
Monitor:	L"x:programmname",yy,aaaa S"x:programmname",yy,aaaa,bbbb
optionale Parameter:	
Dx	= Laufwerksnummer (0/1)
Uy	= Geräteadresse (4..15)
z	= Bank-Nummer (0..15)
aaaa	= Startadresse des Bereichs (0..65535)
bbbb	= Endadresse des Bereichs +1 (0..6535)

Bload ist, wie der Name schon sagt, ein Ladebefehl. Aber es gibt doch schon 'Dload'? Wozu soll ein weiterer Befehl dienen?

Des Rätsels Lösung liegt in der Art und Weise, wie die Daten geladen werden. Bei Dload wird das Programm immer an den Start des BASIC-Speichers geladen, ungeachtet des Bereichs, aus dem es abgespeichert wurde. Bei BASIC-Programmen ist das nicht schlimm. Programme, die in Maschinensprache geschrieben sind, würden allerdings nicht immer laufen. Sie können nur in einem

bestimmten Speicherbereich funktionieren. Auch Grafikbilder müssen oft wieder an die Originalstelle geladen werden.

Deshalb wird beim Speichern immer die Startadresse des Programms mit abgespeichert. Bload lädt das Programm wieder an diese Adresse.

Das Gegenstück zu bload lautet 'bsave'. Damit können beliebige Bereiche abgespeichert werden. Bei Dsave wird immer nur ab dem Beginn des BASIC-Speichers gespeichert.

Der entsprechende Lade- und Speicherbefehl des Monitors hat schon bei Dload bestimmte Bereichsangaben benötigt. Er ist nur speicherbezogen durchführbar. Doch in einem BASIC-Programm können die Monitorbefehle nicht angesprochen werden. Deshalb hat man Bload und Bsave eingebaut.

Diese Floppy-Funktionen sind nämlich nicht wegzudenken. Man benötigt sie, um Sprites oder Grafikbilder in den richtigen Speicherbereich zu laden. Natürlich freuen sich auch die Maschinensprache-Programmierer, daß sie nun Maschinenprogramme einfacher nachladen können. Bei diesen Anwendungen sollten Sie sich aber nicht auf die mit dem Programm abgespeicherte Startadresse verlassen. Geben Sie deshalb möglichst immer den Parameter 'Pa' an. So können Sie sicher sein, daß die Daten in den richtigen Speicherbereich geladen werden. Ansonsten kann es vorkommen, daß das Programm aus einem Bereich abgespeichert wurde, in dem nun wichtige Teile Ihres jetzigen Programms liegen - ein Systemabsturz wäre die Folge.

Eines ist noch wichtig. Müssen im Monitor die Start- und Endadresse des Speicherbereichs hexadezimal angegeben werden, so ist dies in BASIC nur in dezimalen Werten erlaubt. Wenn Sie hexadezimale Angaben verwenden möchten, müßen Sie den Befehl 'dec (" ")' verwenden. Ausserdem funktioniert der Befehl, wie auch Variablen, nur, wenn er eingeklammert wird.

Weiterhin hat der Bsave-Befehl bzw. der Monitor-Save-Befehl den Fehler, den Inhalt der letzten angegebenen Bereichsadresse nicht abzuspeichern. Geben Sie deshalb immer die Endadresse +1 an.

1.2.7 DIRECTORY/CATALOG - DER INHALT DER DISKETTE

BASIC > 3.0:	DIRECTORY Dx ON Uy,"name" CATALOG Dx ON Uy, "name"
Abkürzung: diR / cA	
BASIC < 3.0:	LOAD "$x:name",y : LIST
Monitor:	@y,x:$name
Parameter (optional): Dx : x Uy : y name :	 = Laufwerksnummer (0/1) = Geräteadresse (4..15) Auswahlstring zum Selektieren bestimmter Dateien

Wir haben jetzt mehrmals gespeichert und geladen. Welche Programme sind aber jetzt auf der Diskette? Unter welchem Namen wurde das letzte Programm nochmal abgespeichert? - Ein Inhaltsverzeichnis muß her.

Damit Sie nicht zu Papier und Bleistift greifen müssen, führt die 1570/71-Floppy automatisch ein Inhaltsverzeichnis der auf der Diskette gespeicherten Programme und Dateien. Dieses wird mit 'catalog' oder mit 'directory' aufgerufen. Warum aber zwei Befehle für ein und dieselbe Sache ? Dies ist wohl auch eine Art Tradition (wie Dverify), da im BASIC 4.0 der CBM-8XXX Computer beide Befehle implementiert wurden. Und BASIC 7.0 soll eben zu allen bisherigen COMMODORE-Dialekten kompatibel sein.

Die Parameter sind die üblichen. Nur der "name" ist etwas Neues. Mit dieser Angabe können Sie, wenn Sie sie anführen, bestimmte Dateien auswählen, die angezeigt werden sollen. Dies ist aber nur

mit Joker sinnvoll, von denen Sie in 1.3.9 mehr erfahren. So ist es zum Beispiel möglich, nur die Enträge, deren Name mit 'a' beginnt, aufzulisten. Fehlt die Namensangabe, dann wird das gesamte Inhaltsverzeichnis geliefert.

Nun zum Inhaltsverzeichnis selbst. Schauen wir uns dazu einmal ein Beispiel an:

titel
1
2
3
4
5
blocks

In der invers dargestellten Titelzeile werden Laufwerksnummer, Diskettenname, ID und Diskettenformat angezeigt. Das Laufwerk ist bei der 1571 natürlich immer 0, da es ein Einzellaufwerk ist. Der darauf folgende Diskettenname und die beiden ID-Zeichen wurden beim Formatieren (siehe 1.2.2) festgelegt. Die Kennung '2A' dient nur dazu, zu erkennen, um welches Commodore Format es sich handelt.

Anschließend folgen die Dateieinträge. Zuerst wird dabei die Zahl der von der Datei belegten Blöcke (Sektoren) angegeben. So hat man eine Vorstellung davon, wie groß das Programm oder die Datei ist. Danach folgt der Name des Eintrags und zu guter letzt der Dateityp. Diese Angabe infomiert über die Art des Eintrags, also ob es sich um eine Datei, ein Programm usw. handelt. Die einzelnen Dateitypen werden durch Kürzel dargestellt:

DEL = gelöschter Eintrag (deleted)
PRG = Programm
SEQ = Sequentielle Datei
USR = User-Datei
REL = Relative Datei

Zum Schluß folgt die Angabe, wie viele Sektoren (Blöcke) auf der Diskette noch frei sind.

1.2.8 SCRATCH - PROGRAMME UND DATEIEN LÖSCHEN

BASIC > 3.0:	SCRATCH "name1,name2,..",Dx,Uy
Abkürzung:	sC
BASIC < 3.0:	OPEN 1,y,15,"Sx:name1,name2,..."
optionale Parameter: x	
Monitor:	@y,Sx:name1,name2,..
Parameter:	
name :	bis zu 5 Dateinamen durch Kommata getrennt
Parameter (optional):	
Dx : x	= Laufwerksnummer (0/1)
Uy : y	= Geräteadresse (4..15)

Vielleicht haben Sie vielleicht ein paar Testprogramme abgespeichert und möchten sie wieder löschen. Dazu dient der 'scratch'-Befehl. Er löscht den Eintrag aus dem Directory und gibt die von dem Programm oder der Datei belegten Blöcke wieder zum Beschreiben frei.

Bis zu fünf Einträge können gleichzeitig gelöscht werden. Die Namen der einzelnen Dateieinträge werden durch Kommata getrennt.

Doch wie schnell hat man die RETURN-Taste betätigt und das falsche Programm gelöscht, weil man den falschen Namen angegeben hat. Deshalb fragt der Computer in BASIC 7.0 vorher nochmals nach, ob Sie sich wirklich sicher sind ('Are you sure'). Wenn ja, antworten Sie mit 'y', wie 'yes'. Jede andere Taste führt

zum Abbruch. Wird die Frage positiv beantwortet, dann gibt es allerdings kein Zurück mehr - alle angegebenen Dateien werden gelöscht. Sollten Sie versehentlich den falschen Eintrag entfernt haben, dann hilft nur noch der Diskmonitor. Näheres darüber in Kapitel 6.3.2.

Die Daten des Eintrags werden beim Löschen nämlich nicht überschrieben, sondern nur als gelöscht gekennzeichnet. Soll eine gelöschte Datei wiederhergestellt werden, dürfen Sie nach dem versehentlichen Scratch-Befehl allerdings auf der Diskette nichts mehr abspeichern. Es ist ja möglich, daß dadurch die gelöschten Informationen überschrieben werden - danach gäbe es wirklich keine Rettung mehr.

Nach den Löschen übermittelt die Floppy eine Rückmeldung an den Computer, die dieser in BASIC 7.0 auf dem Bildschirm ausgibt:

01, FILES SCRATCHED, XX, 00

Die Zahl XX gibt an, wie viele Dateien entfernt wurden. Dies ist besonders bei der Verwendung von Jokern interessant, um zu überprüfen, wieviele Einträge vom Löschen betroffen waren.

Der Scratch-Befehl kann natürlich auch in einem BASIC-Programm verwendet werden. Die Frage 'Are you sure' wird aber nur im Direktmodus gestellt. Im Programm entfällt diese Rückfrage. Sie müssen also selbst dafür sorgen, daß Sie nicht die falschen Einträge löschen. Natürlich wird auch keine Rückmeldung an den Computer übertragen. Wenn Sie die Meldung im Programm überprüfen möchten, müssen Sie sie von der Floppy anfordern. Darüber informiert Sie das nächste Kapitel.

Zum Schluß sollten Sie noch eines beachten. Dateien, die im Directory mit einem Stern gekennzeichnet sind, dürfen nicht mit Scratch gelöscht werden. Bei diesen Dateien wurde der Speichervorgang unterbrochen. Deshalb ist die Sektorverkettung nicht in Ordnung. Beim Löschen würden unter Umständen auch noch andere Daten entfernt. Verwenden Sie deshalb bei mit '*' gekennzeichneten Dateien immer den Validate-Befehl.

1.2.9 DS/DS$/ST - WENN EIN FEHLER AUFTRITT ...

BASIC > 3.0:	PRINT DS / PRINT DS$ / PRINT ST
Abkürzung:	? DS / ? DS$ / ? ST
BASIC < 3.0:	10 OPEN 1,y,15 20 GET#1,A$:PRINTA$; :IFST<>64THEN20 30 CLOSE1 RUN
Monitor:	@y

Nur wer nichts macht, macht keine Fehler. Und so wird es auch Ihnen sicherlich bald passieren, daß irgendetwas nicht richtig funktioniert. Stellen Sie sich nur vor, sie wollen ein Programm laden und haben keine Diskette eingelegt. Wie verhält sich jetzt die Floppy? Probieren Sie es doch einfach einmal aus!

Sofort fällt das Blinken der roten (1570) bzw. grünen (1571) Leuchtdiode auf. Befindet sich die Floppy im 1570/71-Modus, dann blinkt die Diode doppelt so schnell wie im 1541-Modus. Es handelt sich also um keine andere Fehlerart. Blinkt die Diode schon nach dem Einschalten, dann hat die interne Selbsttestroutine einen Fehler in der Betriebselektronik gefunden. Hier helfen nur noch die in Kapitel 6.3.6. gegebenen Informationen weiter.

Damit Sie die Ursache für das Blinken erfahren, stellt das Laufwerk eine Fehlermeldung bereit. Diese kann über die Variablen 'ds' und 'ds$' abgefragt werden. Deshalb dürfen Sie diese Variablennamen in Ihren Programmen nicht verwenden. Die Fehlermeldung kann allerdings nur einmal abgefragt werden. Danach erlischt auch die Floppy-Leuchtdiode. Bei der nächsten Abfrage erscheint wieder die 'Ok-Meldung'. Deshalb wird von

BASIC 7.0 die letzte Abfrage in ds/ds$ gespeichert. Mit 'ds$' wird die komplette Meldung auf dem Bildschirm angezeigt.

Schauen wir uns einmal den Aufbau einer solchen Fehlermeldung an:

NN, MELDUNG, TT, SS

Jeder Fehler hat eine Nummer (NN). Dadurch kann die genaue Fehlerursache bestimmt werden. Anschließend folgt der Name des Fehlers im Klartext, z.B. 'Read Error' für einen Lesefehler. Die Angaben TT und SS stehen für Spur- (Track) und Sektornummer der Stelle, an der der Fehler aufgetreten ist. Die genaue Bedeutung der Fehler, die Ursachen und mögliche Abhilfen sind in Kapitel 8.6 aufgeführt.

Wenn Sieanstelle von 'ds$' nur die Variable 'ds' benutzten, erhalten Sie nur eine Zahl - in diesem Fall die Fehlernummer. Dies ist oft bei der Analyse der Rückmeldung hilfreich. Ist ein Befehl fehlerfrei ausgeführt worden, dann gibt Laufwerk eine 'Ok'-Meldung aus, die die Nummer 0 hat. Selbstverständlich blinkt die rote/grüne (1570/1571) Leuchtdiode dann nicht. In Ihren Programmen sollten Sie nach Floppybefehlen immer prüfen, ob 'ds' Null enthät oder ein Fehler vorliegt. Dies kann beispielsweise durch diese Programmfolgen geschehen:

C-64-Modus:

```
10 open 1,8,15 : input#1,a,a$,b,c : close 1
20 if a<>0 then print a;a$;b;c : stop
```

für alle Rückmeldungen (auch scratch)

```
10 open 1,8,15 : input#1,a,a$,b,c : close 1
20 if a>19 then print a;a$;b;c : stop
```

nur für Fehler (scratch ignorieren)

C-128-Modus:

```
10 if ds<>0 then print ds$ : stop
```
für alle Rückmeldungen (auch scratch)

```
10 if ds>19 then print ds$ : stop
```
nur für Fehler (scratch ignorieren)

Neben DS und DS$ gibt es noch eine weitere Variable, die über den aktuellen Systemzustand informiert, die Variable ST. Selbstverständlich dürfen Sie auch diesen Namen nicht für andere Variablen verwenden. Der Begriff 'ST' stammt von 'Status', und genau das ist auch die Funktion von ST. Die Variable informiert über den Status, also den Zustand des Ein-/Ausgabesystems. Die Tatsache, daß es sich dabei hauptsächlich um den Kassettenrekorder handelt, soll nicht weiter stören. Die Bits für die Kassettenanwendung wurden deshalb weggelassen:

Bedeutung der Bits des Statusbytes

Bit	Dez.	Funktion
0	1	Time-out beim Schreiben
1	2	Time-out beim Lesen
6	64	EOI Ende der Daten
7	128	EOT Ende des Blocks

Für den Diskettenbetrieb interessieren nur die Bits 0 und 1 sowie Bit 6 und 7. Bit 6 wird EOI genannt, was 'End of Information' bedeutet, also das Ende der Datenübertragung anzeigt. So wird erkannt, wenn das letzte Zeichen einer Übertragung gesendet wird (siehe ds$ für BASIC <3.0).

Die Bits 0 und 1 zeigen einen Time-out an. Wird ein Gerät, das am seriellen Bus angeschlossen ist, vom Computer angesprochen, dann muß es innerhalb einer bestimmten Zeit antworten. Ansonsten setzt der Computer vorraus, daß das Gerät nicht bereit ist. Wenn also die Zeitspanne ausläuft (Time-Out), werden diese Bits gesetzt. Der Grund für ein Time-Out kann darin liegen, daß das

Gerät nur zum Senden oder nur zum Empfangen von Daten geeignet ist.

Die andere Möglichkeit wäre, daß das Gerät (z.B. eine Floppy) gar nicht angeschlossen ist. In diesem Fall ist das Signal 'EOT' gesetzt. EOT bedeutet 'End of Tape', ist also ein Kassettenstatus, der auf den Floppybetrieb übertragen wurde.

Die Variable ST wird nach jeder Floppyoperation entsprechend korrigiert. Ist das Laufwerk nicht angeschlossen oder ausgeschaltet, so wird Bit 7 von ST gesetzt. In diesem Fall meldet sich der Computer sofort mit 'Device not present'. Erfolgte der vorhergehende Floppybefehl in einem Programm, dann wird dieses beendet - eine ärgerliche Sache. Es ist aber möglich, in einem Programm abzufragen, ob das Laufwerk angeschaltet ist, wie die unten angegebenen Beispiele zeigen. Außerdem kann sogar ermittelt werden, ob eine Diskette im Laufwerk liegt.

C-64-Modus:

```
10 poke 768,185
20 open 1,8,15,"i"
30 poke 768,139
40 ifst and128thenprintchr$(19)"Bitte Laufwerk
   anschalten":close1:goto10
50 input#1,a:close1
60 if a<>0thenprintchr$(19)"Bitte Diskette einlegen  ":
   goto10
```

C-128-Modus:

```
10 trap30
20 open 1,8,15,"i":goto40
30 ifer=5thenprintchr$(19)"Bitte Laufwerk anschalten":
   close1:goto10
40 close1
50 ifds<>0thenprintchr$(19)"Bitte Diskette einlegen  ":
   goto10
```

1.3 SYSTEMBEFEHLE DER FLOPPY

1.3.1 DER BEFEHLSKANAL

Wie Sie in 1.2.1 erfahren haben, kommuniziert der Computer mit der Floppy über spezielle Kanäle. So gibt es natürlich auch für die Fehlermeldungen aus dem vorherigen Kapitel einen eigenen Datenkanal - den Befehlskanal.

Wie der Name schon sagt, ist der Kanal nicht nur für Fehler, sondern vor allem für Befehle zuständig. Alle Floppy-Kommandos außer Load/Save/Open/Close, werden über diesen Kanal übermittelt. Da dies recht umständlich ist (siehe BASIC < 3.0), gibt es in BASIC 7.0 eigene Floppy-Befehle. Diese führen allerdings nichts anderes aus, als die reinen Floppy-Kommandos zusammenzustellen und an das Laufwerk zu übermitteln.

Diese Floppy-Kommandos bestehen immer aus einem Buchstaben, der als Abkürzung für eine Funktion steht, wie z.B. 's' für Scratch. Danach folgt die Laufwerksangabe. Dies ist ein Überbleibsel aus der Zeit der Doppellaufwerke der großen COMMODORE Computer. Die C-1570/71 ist ein Einzellaufwerk und deshalb immer Laufwerk 0. Sollten Sie trotzdem Laufwerk 1 der Floppystation anwählen, so erfolgt eine Fehlermeldung. Ganz sinnlos ist diese Syntax nicht, denn COMMODORE plant auch für den C-128 eine Doppelstation, die 1572 heißen soll. Manchmal ist die Laufwerksangabe aber auch erforderlich, um bestimmte Funktionen anzuwählen (siehe Concat-Befehl).

Müssen weitere Parameter, etwa Dateinamen, angegeben werden, dann folgt als Trennzeichen ein Doppelpunkt, danach die Parameter. Wird die Laufwerksangabe weggelassen, so nimmt die Floppystation immer Nummer 0 an. Dies bedeutet, daß Sie normalerweise bei der C-1570/71 von einer Laufwerksangabe absehen können. Den Doppelpunkt dürfen Sie aber trotzdem nicht vergessen, wenn zum Befehlsbuchstaben weitere Daten übermittelt werden.

In BASIC-Versionen <3.0 müssen die Floppy-Befehle immer über den Befehlskanal an das Laufwerk übermittelt werden (außer Load/Save). Um dem Computer mitzuteilen, daß er einen bestimmten Kanal zur Floppy einrichten soll, dient der 'Open'-Befehl. Damit wird der Kanalbetrieb eröffnet. Für den Befehlskanal lautet dies beispielsweise:

```
OPEN 1,8,15
```

Die erste Ziffer ist dabei eine beliebige Nummer zwischen 1 und 255, mit der der Kanal gekennzeichnet wird. Danach folgt die Geräteadresse der Floppy. In unserem Fall ist das die Standardadresse acht. Die letzte Zahl gibt die Kanalnummer an, hier Nummer 15 für den Befehlskanal. Mehr über den 'Open'-Befehl enthalten die Kapitel 1.2.1, 1.4.1 und 1.5.1.

Jetzt kann man über diesen Kanal Befehle an die Floppy senden. Soll zum Beispiel der Scratch-Befehl ausgeführt werden, dann muß 's:dateiname' übermittelt werden. Dazu dient der 'Print#'-Befehl:

```
PRINT#1,"s:dateiname"
```

In diesem Befehl finden wir auch die 1 vom 'Open'-Befehl wieder. Dadurch, daß wir dem Kanal beim 'Open'-Befehl eine Nummer zuteilen, müssen nicht jedesmal alle Angaben (Geräteadresse, Kanalnummer,usw.) neu genannt werden, wenn man eine Meldung auf einen speziellen Kanal geben möchte - die zugeteilte Kanalnummer genügt.

Zum Schluß noch eine Kleinigkeit, die Sie aber unbedingt wissen sollten. Kommandos an die Floppy dürfen nicht länger als 41 Zeichen sein. Der interne Pufferspeicher der C-1570/71 läßt nicht mehr zu. Bei sehr langen Dateinamen führt dies manchmal, besonders beim Scratch- oder Copy-Befehl, zu Einschränkungen. Es kann dabei einfach die mögliche Dateienzahl nicht ausgeschöpft werden. Grundsätzliche Nachteile entstehen Ihnen dadurch allerdings nicht. Sie müssen höchstens eine Aufgabe in mehrere Teileschritte zerlegen.

1.3.2 COLLECT - ORDNEN DER DISKETTE

BASIC > 3.0:	COLLECT Dx ON Uy
Abkürzung:	collE
BASIC < 3.0: optionale Parameter: x	OPEN 1,y,15,"Vx"
Monitor:	@y,Vx
Parameter: Dx : x = Laufwerksnummer (0/1) Uy : y = Geräteadresse (4..15)	

Der 'Collect'-Befehl bringt die Verzeichnisse auf der Diskette wieder in Ordnung. Im Detail handelt es sich um das Directory und die BAM, das Verzeichnis der belegten und freien Blöcke.

Beim 'Collect'-Befehl wird zuerst die BAM gelöscht. Danach stellt die Floppy zu jedem gültigen Dateieintrag die zugehörigen, von der Datei benutzten Sektoren fest. Diese werden dann in der BAM als belegt gekennzeichnet. Zum Schluss wird die neue BAM auf die Diskette geschrieben. Ausserdem entfernt der 'Collect'-Befehl auch alle ungültigen Einträge aus dem Directory. Doch was sind ungültige Einträge?

Derartige Dateien sind durch einen Stern gekennzeichnet und entstehen immer, wenn der Eintrag nicht abgeschlossen wurde. Dies kann dadurch geschehen, daß eine Datei angelegt aber nicht ordnungsgemäß beendet wurde. Dieser Fall tritt auch ein, wenn ein Programm abgespeichert wird, das größer als der freie Platz auf der Diskette ist. Das Abspeichern wird dann mit einer Fehlermeldung unterbrochen und alle vorher freien Blöcke sind belegt - hier hilft nur der 'Collect'-Befehl.

1.3.3 RENAME - UMBENNEN EINER DATEI IM DIRECTORY

BASIC > 3.0:	RENAME "alt" TO "neu",Dx,Uy
Abkürzung:	reN
BASIC < 3.0:	open1,y,15,"rx:neu=alt"
Monitor:	@y,rx:neu=alt
Parameter:	
alt :	alter Dateiname
neu :	neuer Dateiname
Parameter (optional):	
Dx : x	= Laufwerksnummer (0/1)
Uy : y	= Geräteadresse (4..15)

Mit diesem Befehl können Sie einem bestehenden Directory-Eintrag einen neuen Namen geben. Wie Sie oben im Syntax-diagramm sehen, ist dies nicht sehr kompliziert.

Natürlich eignet sich diese Funktion nicht nur für Schönheitsreparaturen im Directory. Sie ist besonders interessant, wenn von Programmen aus mit Dateien gearbeitet werden soll. Die Dateien können immer den gleichen Namen behalten. Bei Änderungen in der Datei muß man diese unter einem Zwischennamen abspeichern und die alte Datei löschen. Danach wird der Zwischenname in den Namen der alten Datei umgeändert. Mit dieser Methode haben Sie am Ende immer nur einen Dateieintrag. Dies können Sie nicht nur für Dateien sondern auch beim Prorammerstellen verwenden. Auf diese Weise entstehen nicht 100 Versionen des Programms auf der Diskette, sondern immer nur die eine, die aktuellste, und das immer unter dem gleichen Namen.

1.3.4 CONCAT - DATEIEN VERKETTEN

BASIC > 3.0:	CONCAT Dx,"quelle" TO Dy,"ziel" ON Uz
Abkürzung:	cO
BASIC < 3.0:OPEN1,z,15,"cy:ziel=y:ziel,x:quelle" optionale Parameter: x	
Monitor:	@z,Cy:ziel=y:ziel,x:quelle
Parameter: ziel : quelle : Parameter (optional): Dx : x = Dy : y = Uz : z =	 Datei, an die Daten angehängt werden Datei die an 'Ziel' angehängt wird Laufwerksnummer (0/1) Laufwerksnummer (0/1) Geräteadresse (4..15)

Der 'Concat'-Befehl dient zur Ankettung einer Datei an eine bestehende. Dabei werden die Daten der Quelldatei an die Zieldatei angefügt. Die Quelldatei wird nicht gelöscht.

Diese Verkettung funktioniert allerdings nur mit sequentiellen Dateien ('SEQ' oder 'USR'). Programme können auf diese Weise nicht zusammengefügt werden. Das ist ein Problem, das eher im Rechner, als in der Floppy zu lösen ist. Deshalb darf ich hier auf die Bücher zum C-128 verweisen, die im Literaturverzeichnis aufgeführt sind.

Der 'Concat'-Befehl ist eigentlich eine Kopierfunktion und deshalb eine Art Copy-Befehl. Davon mehr im nächsten Abschnitt.

1.3.5 COPY - DATEIEN KOPIEREN

BASIC > 3.0:	COPY Dx,"quelle" TO Dy,"ziel" ON Uz
Abkürzung:	coP
BASIC < 3.0:	OPEN1,z,15 PRINT#1,"Cx:ziel=y:quelle1,quelle2..." CLOSE1
Monitor:	@z,Cx:ziel=y:quelle1,quelle2...
Parameter: ziel : quelle : Parameter (optional): Dx : x = Dy : y = Uz : z =	 Name der neuen Datei Datei(en), aus der neue Datei bestehen soll Laufwerksnummer (0/1) Laufwerksnummer (0/1) Geräteadresse (4..15)

Dieser Befehl kopiert einzelne Dateien. Bei einer Doppelstation erscheint dies sinnvoll - doch was bringt das bei einem Einzellaufwerk, wie der C-1570/71? Natürlich sind die Anwendungsmöglichkeiten des 'Copy'-Befehls etwas eingeschränkt. Es gibt aber durchaus zweckmäßige Einsatzgelegenheiten. Eine davon haben sie schon im vorherigen Kapitel, in Form des 'Copy'-Befehls kennengelernt

Bei einem Einzellaufwerk kann der Copy-Befehl zum Verketten von Dateien benutzt werden. Dabei wird eine neue Datei aus zwei,

drei oder gar vier bereits bestehenden Dateien gebildet. Die Daten der Quelldateien werden in der Reihenfolge, in der die Namen der Quelldateien angegeben werden, zu einer neuen Zieldatei zusammengefügt. Auf diese Art Weise können aber nur sequentielle Dateien verknüpft werden ('SEQ' oder 'USR').

Selbstverständlich dürfen auch Programmdateien kopiert werden. Allerdings ist nur ein Quellprogramm zulässig. Programme können auf diese Weise nicht verknüpft werden. Dies ist ein Problem das man im Rechner lösen muß.

Das Kopieren von Einzeldateien hat trotzdem einen Sinn. Sollen an Dateien Manipulatioen ausgeführt werden, dann sollte dies oft zuerst an Duplikaten erprobt werden. Mit dem 'Copy'-Befehl koennen Sie eine Kopie der Originaldatei erstellen.

Dabei darf ein Aspekt nicht übersehen werden: die Zieldatei muß einen Namen haben, der auf der Diskette noch nicht vorhanden ist. Aber auch hier gibt es einen Sonderfall. Wenn Sie sowohl bei der Quell- als auch bei der Zieldatei die Laufwerksnummer angeben und einen Namen verwenden, der schon auf der Diskette existiert, wird diese Datei von der neuen Zieldatei überschrieben. Nach dieser Methode arbeitet auch der 'Concat'-Befehl.

Wenn Sie ein Programm von einer Diskette auf eine andere kopieren möchten, bediehnen Sie sich spezieller Kopierprogramme. Ein derartiges Programm ist auf der Test-/Demo-Diskette unter dem Namen 'sd.copy.c64' enthalten. Sie müssen dieses Programm im C-64-Modus laden und starten. Es hat allerdings den Nachteil, sehr langsam und unkomfortabel zu sein. Steht Ihnen der Sinn nach Höherem, so darf ich auf die zu diesem Buch erhältliche Diskette hinweisen. Im Anhang sind weitere Informationen darüber aufgeführt.

1.3.6 BACKUP - DISKETTEN DUPLIZIEREN

BASIC > 3.0:		BACKUP Dx TO Dy,Uz
Abkürzung:		baC
BASIC < 3.0:		OPEN1,y,15,"Dy=x"
Monitor:		@z,Dy=z
Parameter:		
Dx :	x	= Laufwerksnummer der Quelldiskette (0/1)
Dy :	y	= Laufwerksnummer der Zieldiskette (0/1)
Uz :	z	= Geräteadresse (4..15)

Dies ist der einzige Befehl des C-128 bzw. der C-1570/71, der überhaupt nicht verwendet werden kann. Backup soll ganze Disketten duplizieren. Dabei wird die Zieldiskette auch gleichzeitig formatiert. Das kann nur bei zwei Laufwerken, also einer Doppelstation, funktionieren. Bei der C-1570/71, einer Einzelstation, ist dies natürlich sinnlos.

Was aber ist zu tun, wenn komplette Disketten kopiert werden sollen? Derartige Sicherheitskopien können mit speziellen Kopierprogrammen erstellt werden. Das auf der Test-/Demodiskette mitgelieferte Programm 'DOS SHELL' beinhaltet beispielsweise ein Kopierprogramm für Backups.

Solch ein Kopierprogramm befindet sich auch auf der Diskette, die zu diesem Buch erhältlich ist. Dieses Programm kopiert nicht nur alle Daten der Diskette. Es entstellt.ein vollständiges Duplikat der Originaldiskette. Auch Lesefehler und Defekte werden übernommen. Dadurch müssen Sie zum Beispiel Reparaturexperimente nicht am Orginal ausführen.

1.3.7 DCLEAR - SCHLIEßEN ALLER KANÄLE

BASIC > 3.0:	DCLEAR Dx, ON Uy
Abkürzung:	dclE
BASIC < 3.0:	nicht vorhanden
Monitor:	nicht vorhanden
optionale Parameter:	
Dx : x	= Laufwerksnummer (0/1)
Uy : y	= Geräteadresse (4..15)

Dieser Befehl schließt alle Kanäle zur Floppy. Dies ist aber lediglich eine rechnerinterne Funktion. Alle entsprechenden Kanaltabellen werden geschlossen. Der Befehl sendet kein 'Kanal schließen' (Close) an die Floppy. Offene Dateien können so nicht beendet werden. Dazu gibt es den Befehl 'Dclose' (Siehe 1.4 und 1.5).

Deshalb hat der 'dclear'-Befehl bei Dateianwendungen kaum einen Sinn. Sie können damit aber eventuelle 'CMD-Kanäle' zur Floppy abbrechen. Dabei handelt es sich um Datenkanäle, auf die die normale Bildschirmausgabe mit CMD umgelenkt wurde. Auf diese Weise wird die Ausgabe anstatt auf den Bildschirm beispielsweise in eine Datei geschrieben.

Wenn Sie den 'Dclear'-Befehl in Ihren eigenen Programmen verwenden, können Sie so sichergehen, daß die Ein- und Ausgabe auf die Standardgeräte erfolgt.

1.3.8 BOOT - CP/M-BERIEBSSYSTEM STARTEN

BASIC 7.0:	boot "Name",Dx,Uy
Abkürzung:	bO
BASIC < 7.0:	nicht vorhanden
Monitor:	gff88c (Sektor 1,0-Boot)

Boot ist ein Befehl mit Doppelbedeutung. Werden Parameter (Name,...) angegeben, dann verhält er sich anders, als wenn diese weggelassen werden. Betrachten wir zuerst die einfachere Version - Boot mit Parametern.

Das Wichtigste dabei ist der Name. Der Computer sucht ein Maschinenprogramm mit diesem Namen im Directory und lädt es in den im Dateieintrag angegebenen Speicherbereich der aktuellen Bank (normalerweise 0). Danach wird die Startadresse angesprungen. Sie müssen also dafür sorgen, daß an dieser Stelle auch etwas sinnvolles steht - ansonsten stürzt das Computersystem ab.

Gibt man nur Boot allein ein, dann liest der Rechner Sektor 0 auf Spur 1. Lauten die ersten drei Zeichen des Sektors 'cbm', dann handelt es sich um einen Autoboot-Sektor. Ansonsten wird der Boot-Befehl beendet.

Der Autoboot-Sektor muß nun eine Reihe von Daten und ein Startprogramm enthalten. Dieses ist dann für weitere Aktionen (Nachladen usw.) verantwortlich. Im folgenden werden die Bedeutungen der Bytes kurz erklärt. Für ausführlichere Studien zum Boot-Befehl sei auf das Buch 128 INTERN verwiesen. In Kapitel 7.7 dieses Buches wird der Befehl ausführlich erklärt. Ausserdem ist auch das dazugehörige Kernal-Listing enthalten.

Aufbau des Boot-Sektors

Byte	Funktion
0-2	'CBM'-Kennung zur Identifikation
3/4	Speicheradresse bei weiteren Boot-Sektoren
5	Speicherkonfiguration bei Folgesektoren
6	Zahl der noch folgenden Boot-Sektoren
ab 7	Boot-Meldung bei 'Booting ...' bis $00
	Name des nachzuladenden Programms bis $00
	eigenes Boot-Programm

Ist eine Boot-Meldung angeben, dann wird diese nach 'Booting' auf dem Bildschirn ausgedruckt. Soll keine Meldung erfolgen, muß trotzdem das Trennzeichen $00 bei Byte 7 abgelegt sein. Danach wird getestet ob weitere Boot-Sektoren geladen werden sollen (Byte 6 ungleich 0). Wenn ja, gelten dafür die Daten aus Byte 3 bis 5.

Im folgenden wird der String, der nach der Boot-Meldung folgt, analysiert. Der Boot-Befehl lädt dann das gleichnamige Programm von der Diskette.

Zuletzt kann dann noch eine eigene Boot-Routine eingebaut werden. Das Programm wird in den Kassettenpuffer im Speicher des Computers geladen und gestartet. Die Boot-Routine darf nie fehlen, da der Computer sonst den Kassettenpuffer anspringt und dort gar nichts sinnvolles steht. Ein Systemabsturz wäre die Folge. Die CP/M-System-Diskette wird ebenfalls mit Boot gestartet. Die Boot-Routine des Sektor 1,0 schaltet dabei den Z-80 an, der das Nachladen von CP/M-Plus organisiert.

Wie Sie sicher schon gemerkt haben wird nach jedem Reset oder Einschalten des Computers automatisch der Boot-Befehl aufgerufen. Ist im Laufwerk eine entsprechende Diskette eingelegt, dann wird diese geladen und das Programm automatisch gestartet.

Diese Art des Ladens bietet sich auch bei professionellen Programmen an.

1.3.9 WILDCARDS UND JOKER

Bisher haben Sie bei allen Floppy-Befehlen den ganzen Namen des Programms oder der Datei, worauf sich der Befehl beziehen soll, angeben müßen. Nehmen wir an, Sie haben verschiedene Programmversionen erstellt, sagen wir 'Test1', 'Test2' usw., und möchten diese alle löschen, dann währen Sie gezwungen, alle Dateinamen anzugeben - eine ziemlich aufwendige Sache.

Aus diesem Grund bietet die Floppy auch die Möglichkeit, Namen abzukürzen oder ganze Namensgruppen gleichzeitig anzusprechen. Die Schlüsselzeichen dabei sind der Stern ('*') und das Fragezeichen ('?'). Deshalb können diese beiden Zeichen auch nicht in Dateinamen verwendet werden. Man nennt sie Joker (bei der C-1570/71) oder auch Wildcards (unter CP/M).

Betrachten wir zuerst die Funktion des Fragezeichens. Das Fragezeichen steht immer als Platzhalter für ein beliebiges Zeichen. Geben Sie also zum Beispiel beim Scratch-Befehl 'Test?' als Dateinamen an, so werden alle Dateien, die mit 'Test' und einem weiteren Zeichen benannt sind, angesprochen. Das Fragezeichen kann natürlich an beliebiger Stelle im Namen stehen. Es dürfen unbegrenzt viele Fragezeichen verwendet werden. Weiter ist es auch erlaubt, nur Jokerzeichen zu gebrauchen. Der Name '????' würde alle Dateien anwählen, die aus vier Zeichen bestehen. Kurzum, fast alle Kombinationen sind möglich.

Wenden wir uns dem zweiten Jokerzeichen zu, dem Stern ('*'). Dieser Joker kürzt den Dateinamen ab. Wird ein Stern angeben, so bedeutet dies, daß der Name noch weitere Zeichen enthalten kann. Es wird einfach die erste Datei im Directory angewählt, auf die die explizit angegebenen Zeichen zutreffen. Wenn Sie nur den Stern als Dateinamen angeben, wird der erste Eintrag des Direc-

tory bearbeitet. Dazu muß vor dem Stern unbedingt eine Laufwerksangabe stehen, wobei der Doppelpunkt völlig ausreicht. Die Floppy nimmt dann Laufwerk 0 an. Der Stern allein, ohne Laufwerksnummer, ist eine Sonderfunktion und wählt das zuletzt aufgerufene (Load/Save/Open) Programm auf. Ist zwischenzeitlich allerdings ein Fehler eingetreten (LED blinkt), dann funktioniert diese Methode nicht mehr.

Es gibt noch ein weiteres Zeichen, das bei der Dateiauswahl hilfreich ist, das Gleichheitszeichen ('='). Mit diesem Zeichen kann die Auswahl der Dateien auf ganz bestimmte Dateitypen eingeschränkt werden. Dazu wird nach dem Dateinamen das Gleichheitszeichen und danach der erste Buchstabe des Dateityps eingegeben. So wählt zum Beipiel 'a*=p' die erste Datei, die mit 'a' beginnt und gleichzeitig ein Programm ist, an.

Zum besseren Verständnis der vielen Kombinationsmöglichkeiten von Jokern und Gleichheitszeichen, hier einige Beipiele:

a*	erster Eintrag, der mit 'a' beginnt.
a*cd	wie oben. Alles nach dem Stern wird nicht beachtet.
a?	alle 2-stelligen Namen, mit erstem Buchstaben 'a'.
???*	alle Einträge mit mindestens drei Buchstaben.
a*=s	alle sequentiellen Dateien die mit 'a' beginnen.

Joker dürfen aber nicht bei allen Floppyfunktionen verwendet werden. Außerdem hat die Anwendung von Jokern bei verschiedenen Befehlen unterschiedliche Auswirkungen. Die untenstehende Tabelle gibt darüber Aufschluß.

Befehl	ist Joker erlaubt?	angewählte Datei
Dload / Bload Dverify	immer	erster identischer Dateiname
Dsave / Bsave	kein Joker	neuer Dateiname
Directory	immer	alle identischen Dateien
Scratch	immer	alle identischen Dateien
Rename Concat / Copy	nur voller Dateiname	nur angegebene Dateinamen

1.4 DIE SEQUENTIELLE DATEI

1.4.1 WAS IST EINE SEQUENTIELLE DATEI?

Datenbanken und Dateien, sind die Begriffe, die heutzutage in aller Munde sind. Nicht nur aufgrund der Probleme mit dem Datenschutz. Auch die neuen Möglichkeiten, die der Computer in der Datenverwaltung gegenüber einer althergebrachten Kartei bietet, sind beachtlich. Natürlich kann auch die C-1570/71 Daten handhaben. Wie das alles im Detail vor sich geht, davon handeln die Kapitel über sequentielle und relative Dateien.

Die Datenaufzeichnung auf der Diskette kann man auch mit einem Buch oder einer Papyrusrolle vergleichen. Die historisch zuerst vorkommende Form der Schriftaufzeichnung war die Papyrusrolle. Bei diesem, noch aus biblischen Zeiten stammenden, Medium werden die Informationen einfach der Reihe nach aufgeschrieben. Sucht man nun etwas bestimmtes, dann muß man die gesamte Rolle von vorne aufrollen und durchsehen, um das

Gesuchte zu finden. Möchte man neue Informationen aufschreiben, geht das nur am Ende des Rollentext. Eingfügungen sind nicht möglich. Diese wohl einfachste Art der Aufzeichnung finden wir auch beim Computer wieder - die sequentielle Datei.

Zum besseren Verständnis nun ein konkretes Beispiel. Nehmen wir an, Sie möchten eine Geburtstagsdatei anlegen. Darin speichern Sie jeweils den Vor- und Zunamen bestimmter Personen und den entsprechenden Geburtstag. Als Beipiel sollen folgende Namen gespeichert werden:

Hans Müller	01.03.1966
Gert Schneider	24.07.1952
Helga Schmidt	02.09.1967

Die Daten werden nun, wie der Name der Dateiform schon sagt, sequentiell abgelegt. Es ist das Wesen dieser Dateiart, das die Daten nur nacheinander abgespeichert und auch nur aufeinanderfolgend gelesen werden können. Hat man also zum Beispiel die drei Personen abgespeichert und möchte später den Geburtstag von Helga Schmidt wissen, dann muss man eben zuerst die Daten von Müller und Schneider überlesen, bis man zu Schmidt kommt. Dabei taucht noch ein weiteres Problem auf. Woher weiß der Computer, wann ein Name, ein Geburtsdatum zu Ende ist? Fragen wir uns dazu doch einfach, woran merkt das der Mensch? Na klar, an den Leerzeichen zwischen den einzelnen Daten. Wir könnten den Computer so programmieren, daß er die Leerzeichen entsprechend beachtet. Heißt aber einer der Personen einmal 'Hans Otto' mit Vornamen, dann funktioniert unser Programm schon nicht mehr einwandfrei, da hier ein störendes Leerzeichen im Vornamen vorkommt.

Also nimmt man einfach ein anderes Zeichen zum Trennen der Daten. Bei den Computerdateien ist das meißten der ASCII-Wert 13. Man kann ihn mit CHR$(13) erzeugen. Sie kennen diesen Wert sicherlich schon. Es ist der gleiche Code, wie ihn auch die Return-Taste hat. Aber auch das Zeichen, das auf dem Bildschirm den Cursor in die nächste Zeile springen läßt (Wagenrücklauf) ist

der ASCII-Wert 13. Das Return-Zeichen wird jetzt immer 'Carrige Return' (engl. = Wagenrücklauf) oder kurz 'CR' heißen.

Eine Folge von Zeichen, die durch ein 'CR' beendet ist, nennt man Datenfeld. Mehrere logisch zusammengehörende Datenfelder ergeben einen Datensatz. In unserem Fall wären Namen und Geburtsdaten einzelne Datenfelder, die zusammen einen Datensatz bilden. Die Datei hätte so drei Datensätze. Die Datensätze kann man dadurch unterscheiden, daß man wiederum ein entsprechendes Trennzeichen verwendet. Normalerweise wird diese Trennung aber von der Programmlogik vorgenommen. Das heißt, das Programm weiß genau, wieviel Datenfelder jeder Satz hat und kennt dadurch auch Satzanfang und Ende, indem es immer mitzählt.

1.4.2 DAS ERÖFFNEN EINER DATEI

Wir wollen jetzt das Beispiel aus dem vorhergehenden Abschnitt in die Tat umsetzen und eine sequentielle Datei aufbauen. Dazu muß man zuerst der Floppy mitteilen, wie die Datei heißen soll, welchen Typs sie sein soll und so weiter. Den dafür nötigen Befehl kennen Sie schon - der Open-Befehl. Schauen wir uns einmal die Syntax des Befehls an:

```
OPEN x,y,z,"a:name,b,c"
```

Die Parameter x,y und z sind die logische Dateinummer, Geräte- und Sekundäradresse, die wir schon in 1.2.1 behandelt haben. 'a:' ist die Laufwerksnummer, die bei der C-1570/71 aber auch weggelassen werden kann. Nun kommt der Name. Hier sind Ihrer Fantasie keine Grenzen gesetzt. Nur, 16 Zeichen darf er nicht überschreiten und er darf auf der Diskette noch nicht vorkommen.

Jetzt kommen die für die Dateiverwaltung wichtigen Angaben. ',b' ist der Dateityp und ',c' die Dateibetriebsart. Bei einer sequentiellen Datei muß der Dateityp 's' heißen. Danach kommt die Betriebsart. Soll die Datei zum Schreiben, also zum ersten mal, angelegt werden, ist hier ein 'w' anzugeben (engl. write = schreiben).

Wie Sie vielleicht bemerkt haben ist oben die BASIC <3.0 Syntax verwendet. Selbstverständlich gibt es hier auch einen komfortableren BASIC 7.0 Befehl. Sie sollten nur einmal die prinzipielle Übermittlungsformen kennenlernen. Denn BASIC 7.0 sendet an die Floppy nichts anderes, als den BASIC 3.0 Befehlsstring. Jetzt der äquivalente BASIC 7.0 Befehl:

DOPEN#x,"name",Da,Ux,b

Wie Sie sicher schon bemerkt haben fehlt der Parameter 'z'. Die Sekundäradresse muß bei Dateien zwischen 2 und 14 liegen. Dies wählt der Computer nun automatisch. Auch die Geräteadresse 'Uz' dürfen bei Laufwerk 8 weggelassen werden. 'Dx' ist sowieso überflüssig, da die C-1570/71 ein Einzellaufwerk ist. Wie Sie sehen reduziert sich der Befehl normalerweise auf:

DOPEN#x,"name",b

Die logische Kanalnummer x können sich sich aber nicht sparen. Dadurch wird bei den einzelnen Ausgaben unterschieden, in welche Datei sie gehen sollen. Verwenden Sie bitte nur Zahlen zwischen 1 und 127. Werte zwischen 128 und 255 haben zur Folge, das nacher bei Ausgaben, nach den übermittelten Daten nicht nur ein 'CR' Trennzeichen, sondern zusätzlich noch der ASCII-Wert 10 (Line Feed) gesendet wird. Und genau dieses stört bei den meißten Anwendungen.

Nun zurück zu unserem Beispiel - der Geburtstagsdatei. Eröffnen Sie nun diese Datei mit:

DOPEN#1,"geb.tag",w

Jetzt tritt die Floppy in Aktion. Zuerst wird geprüft ob der Dateiname schon auf der Diskette vorhanden. Dies würde einen 'File Exists Error' hervorrufen. Darum ist es immer angebracht, nach dem Dopen-Befehl die Fehlervaiable ds zu prüfen. Ist sie 0, dann kann man sicher sein, daß das Öffnen der Datei erfolgreich war.

Nun leuchtet die rote (1570) bzw. grüne (1571) Laufwerksanzeige auf solange die Datei geöffnet ist. Sie weist Sie darauf hin, daß Sie die Diskette dem Laufwerk zwischenzeitlich nicht mehr entnehmen dürfen. Die Floppy wartet nun auf die Daten vom Computer. Die Anzeige erlischt erst dann wieder, wenn die letzte aktive Datei wieder beendet wurde.

1.4.3 DAS SPEICHERN VON DATEN

Nachdem die Datei erfolgreich eröffnet wurde, müssen wir jetzt die einzelnen Datenfelder in die Datei schreiben. Dazu diehnt der gleiche Befehl, wie der, der auch Informationen auf dem Bildschirm ausgibt - der 'Print'-Befehl.

Allerdings ist dieser für die Ausgabe über Datenkanäle etwas modifziert worden, schließlich muß man ja auch wissen, über welchen Kanal die Daten gehen sollen. Er heißt deshalb:

```
PRINT#x,"daten"
```

Die Abkürzung für den Print-Befehl ist aus diesem Grund auch nicht das Fragezeichen, sondern 'pR'. Dies müssen Sie beim Eingeben von Programmen beachten, wenn Sie von den Abkürzungen Gebrauch machen. Unsere Geburtstags datei würde so etwa folgende Befehlsfolge enthalten:

```
10 DOPEN#1,"geb.tag",w                  Datei eröffnen
20 INPUT "vorname ";a$                  Daten in Variable
30 INPUT "name ";b$                     eingeben
40 INPUT "geburtstag ";c$
50 PRINT#1,a$;CHR$(13);b$;CHR$(13);c$   Daten in Datei
                                        schreiben
```

Wie Sie sehen, müssen wir die einzelnen Datenfelder durch ein 'CR', das durch chr$(13) erzeugt wird, abtrennen. Aber warum fehlt es nach dem letzten Datenfeld, dem Geburtstag? Ganz einfach, der Print-Befehl sendet dieses immer automatisch. Bei lo-

gischen Kanalnummern über 127 würde er zusätzlich sogar noch ein 'line feed' übermitteln. Es gibt allerdings auch Fälle in dehnen dieses automatische Senden unerwünscht ist. Dann müßen Sie die Print-Zeile einfach mit einem Semikolon (';') abschließen. Der Computer weiß dadurch, daß er kein CR ausgeben soll.

1.4.4 SCHLIEßEN DER SEQUENTIELLEN DATEI

Sind alle Daten in die Datei eingetragen, dann dürfen Sie jetzt die Diskette nicht einfach wieder herausnehmen. Schließlich muß die Floppy noch die Sektorenverkettung beenden, die benutzten Blöcke abzählen und im Directory vermerken. Ist das alles erfolgreich geschehen, dann wird Directory-Eintrag der Datei gekennzeichnet, daß die Datei ordnungsgemäß abgeschlossen wurde.

Diese Funktion wird mit dem 'Close'-Befehl aufgerufen. Seine Syntax lautet:

DCLOSE#x ON Uy

x ist wieder die logische Nummer des Datenkanals zur Datei. Mit y kann das Gerät ausgewählt werden. Bei der Geräteadresse 8 darf die Angabe entfallen. Aber auch die Kanalnummer ist nicht immer erforderlich. Der Befehl Dclose allein schließt alle momentan geöffneten Dateien, wobei der C-64/C-128 maximal 10 Stück gleichzeitig verwalten kann. Diese Anzahl werden Sie kaum ausnutzen, denn auf einem Floppylaufwerk können nicht mehr als drei sequentielle Dateien gleichzeitig bearbeitet werden.

In BASIC <3.0 existiert nur ein Befehl, um eine ganz bestimmte Datei zu schließen. Doch hier gibt es einen Trick. Schließt man den Befehlskanal, dann werden von der Floppy automatisch auch alle anderen Kanäle und Dateien geschlossen. Öffnen Sie deshalb am Anfang Ihrer BASIC-<3.0-Programme einfach den Befehlskanal. Wenn Sie ihn dann wieder schließen, hat das die gleiche Wirkung wie 'Dclose'.

Sollten Sie das Schließen der Datei vergessen haben, dann sind die Daten trotzdem noch nicht verloren. Wie Sie sie wieder retten können, erfahren Sie im nächsten Abschnitt.

1.4.5 ABRUFEN AUS EINER DATEI

Das alleinige Speichern von Daten ist natürlich uninteressant. Sie wollen ja schließlich mit den Daten etwas anfangen. Deshalb gibt es auch mehr Befehle zur Verarbeitung von Daten aus Dateien, als zum Schreiben von Daten in Dateien.

Zur Verarbeitung von Daten müssen Sie die Datei zuerst wieder eröffnen, jetzt natürlich zum Lesen. Dabei gibt es zwei verschiedene Modi, zum einen den ganz normalen Lesevorgang und zum zweiten einen Spezialmodus, mit dem Sie nicht ordnungsgemäß geschlossene Dateien wiederherstellen können.

Um normale sequentielle Dateien zu lesen, muß als Betriebsmodus ein 'r' (engl. read = lesen) angegeben werden. Dies führt der Computer immer selbsständig aus, wenn sie kein 'w' nennen. Die Geburtstagsdaten stehen folglich nach

```
DOPEN#1,"geb.tag"
```

wieder zur Verfügung. Um die Daten jetzt in den Computer einzulesen, gibt es zwei Möglichkeiten. Den GET- und den INPUT-Befehl. Der Einfachere davon ist der 'Get'-Befehl, weshalb wir ihn auch zuerst besprechen wollen.

Dieser Befehl wurde, genau wie sein Gegenstück 'Print', für die Dateiverwaltung etwas modifiziert. So muß man hier die zugeordnete Kanalnummer angeben. Die Syntax des Befehls lautet:

```
GET#x,a$
```

Genauso wie der normale 'Get'-Befehl ein Zeichen von der Tastatur annimmt, wird hier ein Zeichen aus der entsprechenden Datei gelesen. Dabei beginnt die Floppy am Anfang der Datei und übermittelt so Zeichen für Zeichen bis zum Ende der Datei.

Irgendein beliebiges Zeichen der Datei können Sie auf diese Weise nicht auslesen, womit Sie nun auch schon einen Nachteil der sequentiellen Datei kennenlernen.

Auf unser Beipiel angewandt könnte man auf folgende Art und Weise die einzelnen Datenfelder wieder einlesen:

```
10 DOPEN#1,"geb.tag"
20 GET#1,z$:a$=a$+z$:IFz$<>CHR$(13)THEN20
30 GET#1,z$:b$=b$+z$:IFz$<>CHR$(13)THEN30
40 GET#1,z$:c$=c$+z$:IFc$<>CHR$(13)THEN40
50 PRINT "name : ";a$
60 PRINT "vorname : ";b$
70 PRINT "geburtstag : ";c$
```

Im obigen Programm werden so lange Zeichen von der Diskette gelesen, bis ein Trennzeichen 'CR' auftritt. Alle bis dahin gelesenen Zeichen (incl. CR), sind dann einer Stingvariable zugeordnet, die weiterverarbeitet werden kann (a$,b$,c$). Mit dieser Methode lassen sich die einzelnen Datenfelder trennen.

Doch wie erfährt man, wann das letzte Datenfeld, der letzte Datensatz der Datei gelesen wurde? dazu dient die in Kapitel 1.2.9 besprochene Statusvariable ST. Bit 6 dieser Systemvariablen hat den Wert 1, wenn ein EOI-Signal übermittelt wurde. Dieses EOI bedeutet nichts anderes als 'End of Information' (Ende der Information). Das Bit zeigt uns also an, wann das letzte Zeichen gesendet worden ist. Der Test, ob dieses Bit den Wert 1 hat könnte etwa so aussehen:

```
IF ST AND 64 THEN ...
```

Achten Sie unbedingt darauf, daß Sie zwischen ST und dem 'And'-Befehl ein Leerzeichen setzen, sonst würde der Computer die Zeile als 's TAN d' interpretieren, und den Tangens benötigen wir in diesem Fall nicht. Der 'If'-Befehl verzweigt in der obigen Form immer, wenn das letzte Zeichen gesendet wurde. Wollen Sie eine Schleife programmieren, die beim letzten Zeichen verlassen

wird, also dann verzeigt, wenn EOI nicht gesetzt ist, muß die Zeile folgendermaßen lauten:

```
IF NOT ST AND 64 THEN ...
```

Eine Sache müssen Sie noch beachten. Der 'Get'-Befehl liest alles, auch Steuerzeichen, mit einer einzigen Ausnahme, der ASCII-Wert 0. Er wird nicht übermittelt. Ist also ein Zeichen gleich CHR$(0), dann wird nichts gesendet. In solch einem Fall erhalten Sie einen leeren String. Dieses Verhalten des Computers müssen Sie beim Programmieren im Auge halten. Es empfiehlt sich deshalb immer folgender Schritt:

```
GET#1,a$:IFa$=""THENa$=CHR$(0)
```

oder

```
GET#1,a$:a$=LEFT$(a$+CHR$(0),1)
```

Mit diesem einen kleinen Nachteil ist der 'Get'-Befehl gegenüber seinem Kollegen, dem 'Input'-Befehl, geradezu ein Musterknabe. Beim 'Input'-Befehl, den wir nun besprechen wollen, häufen sich die Sonderfälle und Fehlermöglichkeiten.

Auch dieser Befehl wurde für die Datenkanäle angepasst und lautet nun:

```
INPUT#1,a$
```

Dieser 'Input'-Befehl verhält sich genau wie der 'Input'-Befehl, der Eingaben vom Bildschirm entgegennimmt. Natürlich dürfen auch hier die Eingaben nicht endlos lang sein. Schließlich kann ein String nur bis zu 255 Zeichen aufnehmen. Wie Sie aber beim normalen Input schon festgestellt haben, ist immer sehr viel früher Schluß. Das hängt damit zusammen, daß der Computer jede Eingabe über Input, bevor er sie weiterverarbeitet, in einem Puffer zwischenspeichert. Und dieser ist eben nur 88 Zeichen (beim C-64) oder 160 Zeichen (beim C-128) groß. Werden mit dem 'Input#'-Befehl mehr als 88 bzw. 160 Zeichen eingelesen, dann erfolgt eine Fehlermeldung. Diese lautet logischerweise 'String too long'. Wir müssen also unbedingt vermeiden, daß dieser

Fehler auftritt, da sonst der Computer auch das Programm abbricht. Daher ergibt sich natürlich die Frage, wie lange eine Input#-Anweisung eigentlich Zeichen von der Datei einliest?

Dies funktioniert genau wie beim Bildschirm-Input. Wenn der CR-Code gesendet wird, sie also beim normalen Input die RETURN-Taste betätigen, beendet der 'Input'-Befehl die Eingabesequenz. Das Problem besteht also darin, daß einzelne Datenfelder nicht größer als 87 bzw. 159 Zeichen sein dürfen und am Ende ein CR stehen muß. Diese Vorgabe einzuhalten, ist Sache des Programms. Sie als Programmierer, müssen dafür sorgen, daß die Datenfelder nicht länger werden.

Doch damit ist nicht genug. Das Betriebssystem legt Ihnen noch weitere Steine in den Weg. Das sind die Zeichen ':', ',' und ';'. Diese werden in BASIC normalerweise dazu verwendet, Befehle und Parameter voneinander zu trennen. Nichts anderes führt auch der 'Input#'-Befehl aus, wenn er auf diese Zeichen stößt. Kommt in einem Datenfeld ein Doppelpunkt, Komma oder Semikolon vor, dann verhält sich der 'Input#'-Befehl so, als ob er ein CR gelesen hätte - er schließt die Eingabe ab und nimmt an, daß das Datenfeld hier zu Ende sei. Und als Zugabe des Betriebssystems erhalten Sie auch noch einen 'Extra ignored Error'.

Sie müssen aber nicht nur auf diese drei Zeichen achtgeben. Wenn Sie mit INPUT# Zahlenwerte in Variablen einlesen, also zum Beispiel durch 'INPUT#1,a', kann es Probleme geben. Dies ist immer der Fall, wenn noch andere Zeichen als Ziffern im zu lesenden Datenfeld stehen. Der Computer reklamiert das sofort mit 'File Data Error'. Sie können dies nur verhindern, indem Sie in Ihrem Programm konsequent darauf achten, daß Sie ein Datenfeld aus dem gleichen Variablentyp abgespeichert haben, in den Sie es nachher wieder einlesen möchten.

Sicherlich werden Sie sich jetzt schon gefragt haben, wozu es bei all diesen Nachteilen überhaupt einen 'Input#'-Befehl gibt? Die Anwort ist ganz einfach - er ist schneller als der 'GET#'-Befehl. Die Ursache dafür liegt darin, daß vor jeder Meldung an die Floppy das Gerät neu adressiert werden muß. Beim Get#-Befehl findet dies bei jedem Zeichen statt, während die Prozedur beim 'Input#'-Befehl nur einmal pro String erforderlich ist. Trotzdem

können Sie auf den 'GET#'-Befehl nicht verzichten. Er wird immer benötigt, wenn der Input#-Befehl versagen würde.

Nun soll noch auf den Modify-Modus eingegangen werden. Dieser dient dazu, daß nicht ordnungsgemäß geschlossene Dateien noch gelesen werden können. Dies sind all die Dateien, die im Directory mit einem Stern gekennzeichnet sind. Um derartige Dateien zu retten, eröffnet man diese beispielsweise mit:

```
OPEN 1,8,2,"datei,s,m"
```

Wie Sie sehen handelt es sich hier um den BASIC <3.0-Befehl. Das liegt daran, daß BASIC 7.0 keinen Modify-Modus kennt.

Um Daten aus einer nicht ordnungsgemäß geschlossenen datei zu retten, öffnen Sie eine neue Datei, lesen die Daten im Modify-Modus aus der nicht geschlossenen Datei in die neue ein und schließen diese dann ordnungsgemäß ab (Close).

Das Ende der nicht ordungsgemäß geschlossenen Datei kann, wie bei normalen Dateien, mit der Statusvariablen ST erkannt werden. Dabei gibt es allerdings ein Problem. Dadurch, daß die Datei nicht geschlossen wurde, wurde auch die Endekennung bei der Sektorverkettung nicht angelegt. Aus diesem Grund werden beim Lesen mit Sicherheit mehr Daten geliefert, als eigentlich geschrieben wurden. Diese Daten stammen von irgendwelchen sonstigen Sektoren, die zufällig in die Sektorenfolge eingekettet sind. Hier hilft nur ein Nachbearbeiten der Daten "von Hand".

Zum Schluß müssen Sie die nicht geschlossene Datei löschen. Dies funktioniert nur mit dem 'Collect'-Befehl (1.3.2). Der Scratch-Befehl würde aufgrund der fehlerhaften Sektorverkettung eventuell die falschen Sektoren zum Beschreiben freigeben.

1.4.6 ANFÜGEN VON DATEN

Mit einem einmaligen Abspeichern ist es bei einer Datei normalerweise nicht getan. Immer wieder kommen neue Daten dazu, die an die sequentielle Datei angefügt werden müssen. Dazu müßte man die gesamte Datei in den Rechner einlesen und dann als neue Datei wieder abspeichern. Dabei würden die neuen Daten am Ende mit abgespeichert.

Dieses Verfahren ist sehr zeitaufwendig und nicht besonders ausbaufähig. Deshalb gibt es hier eine spezielle Floppyfunktion. Wird eine Datei mit dem Betriebsmodus 'a' (engl. append = anhängen) eröffnet, können Daten angefügt werden. In BASIC 7.0 gibt bietet dafür einen eigenen Befehl:

```
APPEND#x,"name",Dy,Uz
```

Die Angaben x,y und z stellen die Kanal-, Laufwerks- und Gerätenummer dar, die Sie sie schon von den anderen Befehlen kennen. Der "name" ist der Name der Datei, die Sie erweitern möchten. Er kann auch mit Jokern angegeben werden. Die Floppy wählt dann einfach den ersten passenden Eintrag. Bei unserer Beispielanwendung würde der Befehl folgendermaßen lauten:

```
APPEND#1,"geb.tag"
```

Alle Daten, die nun, wie in Kapitel 1.4.3 erklärt, in die Datei geschrieben werden, werden an das Ende der bestehenden Datei angehängt. Aber auch hier gilt wieder 'keine Freud ohne Leid', denn die Floppy hat beim Aufruf der 'Append'-Funktion die Angewohnheit, der Datei mindestens einen Block mehr zuzuweisen. Dies geschieht allerdings nur bei der Blockangabe im Directory. Das bedeutet, daß die Anzahl der benutzen Blöcke aus dem Directory mit der tatsächlich von der Datei belegten Blöcke nicht übereinstimmt. Dies können Sie auch dadurch feststellen, daß alle Blockangaben des Directory addiert, einen anderen Wert ergeben, als wenn man die Anzahl der freien Blöcke von der Gesamtkapazität, insgesamt 664 Blöcke, abzieht.

Doch ganz unproblematisch ist auch der 'Append#'-Modus nicht. Man muß ihn genau wie den 'Dopen'-Befehl mit Close beenden, damit die Datei richtig abgeschlossen wird. Hat man das vergessen, dann sind nicht nur die angefügten Daten verloren, sondern die gesamte Datei. Dieser Fall tritt auch ein, wenn beim Anfügen der Platz auf der Diskette nicht mehr ausreicht und die Floppy 'Disk Full' meldet.

Aus diesem Grund sollten Sie sich den 'Concat-Befehl' aus 1.3.4 wieder in Erinnerung rufen. Mit diesem Befehl wird eine sequentielle Datei an eine bestehende Datei angehängt.

Der Vorteil besteht darin, daß die alte Datei zuerst überhaupt nicht "angetastet" werden muß. Sie speichern die neuen Daten einfach in eine Zwischendatei, die beispielsweise den Namen 'temp' (wie temporär) erhählt. Das Speichern funktioniert genauso, wie wir es in den Kapiteln 1.4.2 bis 1.4.4 behandelt haben.

Sind die neuen Daten in der 'temp'-Datei abgelegt, so wird diese an die bestehende Datei angehängt. In der Praxis könnte das zum Beispiel so aussehen:

```
10 DOPEN#1,"temp",w

   ... Programmteil um Daten in Datei
   zu schreiben ...

50 DCLOSE#1
60 CONCAT "temp" TO "geb.tag"
70 SCRATCH "temp"
```

Diese Methode ist wesentlich sicherer als ein 'Append'-Vorgang. Der einzige Nachteil besteht darin, daß auf der Diskette noch soviel Platz sein muß, daß sowohl die alte Datei 'geb.tag' und die Zwischendatei 'temp' als auch die genauso große neue Datei 'geb.tag' untergebracht werden können. Der Grund dafür liegt darin, daß zuerst die neue Datei 'geb.tag' abgelegt und danach erst die alte Datei 'geb.tag' gelöscht wird. Zuletzt kann man noch 'temp' entfernen, da die Daten von 'temp' jetzt in 'geb.tag' enthalten sind.

1.4.7 ANWENDUNG DER SEQUENTIELLEN DATEI

Wie Sie in 1.4.1 schon erfahren haben, ist die sequentielle Datei die einfachste Form der Datenspeicherung. Die Daten werden nacheinander, das heißt sequentiell abgelegt. Genauso werden die Daten dann wieder eingelesen. Durch die Art der Datenspeicherung ist auch die Anwendung der sequentiellen Datei bestimmt.

Eine Anwendungsmöglichkeit ist beispielsweise der CMD-Kanal. Mit der CMD-Anweisung kann die normale Bildschirmausgabe auf einen Datenkanal umgelenkt werden. Und warum sollte das nicht auch einmal ein Kanal zu einer sequentiellen Datei sein? Auf diese Weise können Sie Programmlistings in eine Datei ausgeben. Diese wiederum wird dann vielleicht von Ihrem Textprogramm verarbeitet o.ä..

Die sequentielle Datei ist immer sinnvoll, wenn man Daten nur zwischenspeichern muß oder es nicht erforderlich ist, auf die Daten wahlfrei zuzugreifen. Genau dies wäre aber notwendig. Bei der sequentiellen Speicherung muß die ganze Datei gelesen werden, bis man den richtigen Eintrag gefunden hat. Bei großen Dateien kann das sehr lang dauern.

Eine Alternative bestände also darin, zuerst die gesamte Datei in den Computer einzulesen. Dann könnte man beliebig auf die Daten zugreifen, wenn man sie in indizierten Feldern abgelegt hat. Ausserdem verläuft der Zugriff auf Variablen im Computer schneller, als das Lesen von der Diskette. Dieses Gebiet der Datenverarbeitung würde allerdings den Rahmen des Floppy-Buchs sprengen. Ich darf deshalb auf die im Literaturverzeichnis aufgeführten BASIC-Lehrbücher verweisen.

Die Computerspeichermethode hat wiederum einen großen Nachteil. So groß der Speicher auch sein mag, er ist nicht unerschöpflich. Damit wäre die maximale Größe einer Datei durch den Speicherplatz im Computer eingeschränkt.

1.5.1 WAS IST EINE RELATIVE DATEI?

Wie Sie in den vorhergehenden Abschnitten gesehen haben, ist die sequentielle Datei eine praktische Sache, doch optimal ist sie nicht. Eine wirklich leistungsfähige Dateiform müßte folgende Merkmale bieten:

- jeder Datensatz ist direkt anwählbar
- einzelne Datensätze können gelöscht werden
- die Datei muß nicht eingelesen werden
- beliebiges Lesen und Schreiben

Von derartigen Fähigkeiten müssen Sie nicht träumen, denn die Floppy C-1570/71 bietet hier die relative Datei. Was das genau ist und wie es funktioniert, das ist Gegenstand der nächsten Abschnitte.

Kehren wir einmal zu unserem Beispiel aus 1.4.1 zurück. Da wurde die sequentielle Datei mit einer Papyrusrolle verglichen. Überträgt man dieses Beispiel auf die relative Datei, dann könnte man diese mit einem leeren Buch vergleichen. Sie können in diesem Buch auf jeder Seite einen Datensatz eintragen. Das Buch kann man auf jeder beliebigen Seite aufschlagen. Auf diese Weise können Sie auf einen beliebigen Datensatz zugreifen. Genauso verhält es sich auch mit einer relativen Datei.

Bei der relativen Datei müssen Sie vor dem Anlegen genau definieren, wie lang jedes Datenfeld und somit jeder Datensatz sein soll. Möchte man beispielsweise auf den 2301. Datensatz zugreifen, dann ist das kein Problem. Die Floppy kann anhand der festen Datensatzlänge und der Nummer des Datensatzes die Position in der Datei errechnen. Im Grunde genommen ist auch beim Buch der Datensatz bereits vordefiniert. Jede Seite kann nämlich eine ganz bestimmte Zeichenmenge aufnehmen. Normalerweise nutzen wir aber nie die ganze Seite des Buches aus, wenn man jedem Datensatz eine eigene Seite zuweist.

Jetzt kennen Sie das Dilemma der Datenverwaltung. Entweder speichern wir Daten sequentiell ab und nutzen die Diskettenkapazität optimal aus, da zwischen den einzelnen Datensätzen kein

Platz verschenkt wird. Oder wir definieren eine feste Datensatzlänge. Damit wird meistens viel Platz auf der Diskette geopfert. Dafür kann man aber jeden Datensatz anwählen, da man seine Position errechnen kann. Im Normalfall wird man der besseren Zugriffsmöglichkeit den Vorrang geben, denn der Speicherplatz spielt bei den heutigen Disketten- oder Festplattenkapazitäten meistens keine so entscheidende Rolle.

Nun wieder zu einem handfesten Beispiel, der Geburtstagsdatei. Wie bereits gesagt, muß jeder Datensatz eine feste Länge erhalten. Dazu gibt man den einzelnen Datenfeldern des Datensatzes vorbestimmte Längen. Die Längen sollten so gewählt werden, daß die Daten im Normalfall gut untergebracht werden können. Natürlich kann es trotzdem Fälle geben, in denen zum Beipiel der Name sehr lang ist und nicht in das Datenfeld paßt. Im Durchschnitt werden die einzelnen Datensätze aber nie vollständig ausgenutzt. Zwischen diesen beiden Extremen müssen Sie Ihre Datensatzlänge wählen. Bei unserer Geburtstagsdatei ist das beispielsweise:

Vorname	15 Zeichen
Datum	10 Zeichen
insgesamt	40 Zeichen

Vielleicht sind Sie in Ihrem Bedienungshandbuch schon auf die Angabe 'max. Größe' gestoßen. Hier heißt es, daß eine relative Datei maximal 167132 Bytes groß sein darf - auch wenn die Diskette doppelseitig verwendet wird. Das hängt mit dem Floppy-Betriebssystem zusammen. Dieses kann nämlich keine größeren relativen Dateien verwalten.

Wenn wir die gesamte Kapazität der relativen Datei ausnutzen, können wir insgesamt 4178 Datensätze zu je 40 Zeichen abspeichern.

1.5.2 DAS ERÖFFNEN EINER DATEI

Nun wieder zur Praxis. Wir wollen eine relative Datei einrichten. Dies ist nicht schwieriger, als bei einer sequentiellen Datei. Der einzige Unterschied besteht darin, daß man die Datensatzlänge definieren muß. Zugelassen sind Werte zwischen 2 und 254 Zeichen. Das Festlegen geschieht mit dem Betriebsmodus 'l'(von engl. length = Länge). Dadurch weiß der Computer, daß es sich um eine relative Datei handelt.

Ob sie die Datei zum Lesen oder Schreiben öffen, ist nicht mehr wichtig. Bei relativen Dateien gibt es diese Unterscheidung nicht mehr. Sie können also nach Herzenslust Einträge überschreiben, anfügen oder auslesen. Der 'Dopen#'-Befehl für unsere Geburtstagsdatei lautet:

```
DOPEN#1,"geb.tag",L40
```

Dabei wird wieder der Eintrag im Directory angelegt. Wenn die relative Datei bereits auf der Diskette existiert, kann die Längenangabe 'L40' entfallen. Sie darf aber durchaus auch angegeben werden. Sie muß nur immer mit der Datensatzlänge, die beim Eröffnen definiert wurde, übereinstimmen.

Weiter sollten Sie sich schon beim Eröffnen der relativen Datei überlegen, wieviele Datensätze sie etwa enthalten wird. Dann wählen Sie den größten zu erwartenden Datensatz an und beschreiben ihn mit CHR$(255). Durch diese Prozedur legt die Floppy nun auch alle vorhergehenden Datensätze an. Dieser Vorgang kann durchaus einige Minuten in Anspruch nehmen.

Der Vorteil besteht darin, daß Sie sicher sein können, daß der Diskettenplatz nicht anderweitig belegt wird und Ihre relative Datei später eventuell keinen Platz mehr hat. Außerdem ist die Anlegeprozedur nachher nicht mehr erforderlich.

1.5.3 DAS SPEICHERN VON DATEN

Zuerst einmal benötigen wir einen Befehl, mit dem wir auswählen, auf welchen Datensatz sich die Schreiboperation beziehen soll. Dieser Befehl lautet:

RECORD#x,y,z

Record kommt aus dem Englischen und bedeutet nichts anderes als 'Datensatz'. Um einen bestimmten Datensatz ansprechen zu können, erhalten die Datensätze einfach fortlaufende Nummern von 1 bis maximal 65535. Diese riesige Spanne werden Sie kaum benötigen. Jede relative Datei, deren Datensätze mehr als zwei Zeichen groß sind, kann dieses Maximum nie erreichen - die Diskettenkapazität ist vorher erschöpft.

Doch nun zu den Parametern des 'Record#'-Befehls. Die Zahl x ist die logische Kanalnummer, also genau dieselbe Nummer wie auch beim 'Print#'- oder 'Input#'-Befehl. Danach folgt die Datensatznummer. Zuletzt kann man noch die aktuelle Position im Datensatz festlegen. Eine Schreib- oder Leseoperation würde dann an dieser Stelle beginnen. Sie können diese Funktion auch dazu benutzen, den Positionszeiger auf ein Datenfeld innerhalb des Datensatz zu richten.

Zum Speichern der Daten steht Ihnen wieder der 'Print#'-Befehl zu Verfügung. Dieser ist genau wie bei den sequentiellen Dateien (siehe 1.4.3) zu handhaben. Das einzige, worauf sie achtgeben müssen, ist, daß sie beim Schreiben nicht mehr Daten angeben, als noch in den Datensatz passen. Sollten Sie trotzdem versuchen, über das Datensatzende hinaus zu schreiben, werden diese Daten ignoriert und die Floppy reklamiert mit '51 Overflow in Record'. Prüfen Sie also gegebenenfalls die Fehlervariable 'ds'.

1.5.4 SCHLIEßEN DER RELATIVEN DATEI

Im Gegensatz zur sequentiellen Datei ist der 'Dclose'-Befehl bei relativen Dateien nicht so "lebensnotwendig". Zumindest sind ihre Daten nicht sofort verloren, wenn Sie den 'Dclose'-Befehl einmal vergessen.

Die Sektorverkettung der relativen Datei wird beim Eröffnen bzw. beim Erweitern der Datei angelegt. Diese kann man also durch ein fehlendes Schließen nicht durcheinanderbringen.

Auch die von der Datei benutzen Blöcke werden immer im entsprechenden Verzeichnis der Diskette belegt. Ein versehentliches Überschreiben durch andere Dateien ist also hier nicht möglich.

Aus diesen Gründen ist eine relative Datei im Directory nie mit einem Stern markiert. Sie ist ja immer voll funktionsfähig.

Trotzdem sollten Sie jetzt das 'Dclose' nicht prinzipiell weglassen. Es hat durchaus noch eine Funktion. Wird die Datei geschlossen, dann stellt die Floppy die Zahl der belegten Blöcke fest und aktualisiert den Directory-Eintrag.

Der Sinn des Ganzen besteht also nicht darin, sich nun immer Dclose zu sparen, weil Sie vielleicht auf die Blockangaben verzichten können. Vielmehr sollten Sie nur wissen, daß die relative Datei fehlertoleranter ist.

Darüber hinaus meldet sich die Floppy im Zusammenhang mit relativen Dateien nicht nur bei der kleinsten Fehlbedienung, sondern stellt auch Meldungen zur Verfügung, wenn die Datei erweitert wird, der Datensatz nicht existiert, die Diskette voll ist usw..

1.5.5 ÄNDERN EINES RECORDS

Daten sind meistens eine sehr kurzlebige Sache. Deshalb ist es wohl am wichtigsten, daß die Informationen in einer Datei nachträglich geändert werden können. Bei der sequentiellen Datei ist das sehr umständlich oder mit einem vertretbaren Aufwand gar nicht möglich.

Die relative Datei kennt in diesem Punkt keine Einschränkungen. Lese- und Schreiboperationen dürfen Sie beliebig verwenden. Sie müssen zum Ändern von Daten nur noch mit dem 'Record#'-Befehl Datensatz und Position der Änderung bestimmen. Mit Print# können Sie das Datenfeld oder den Datensatz überschreiben.

Beim 'Print#'-Befehl müssen Sie unter Umständen nur beachten, nach den Daten ein Semikolon anzugeben. Damit wird die Ausgabe von CR unterdrückt, das ansonsten auch in den Datensatz geschrieben würde.

1.5.6 ANFÜGEN NEUER RECORDS

Eine relative Datei kann, bis zur maximalen Größe von einer Diskettenseite beliebig erweitert werden. Sie müßen dazu nur den benötigten Datensatz mit Record# ansprechen und beschreiben.

Besonders wichtig beim Erweitern ist die Fehlermeldung '50 Record not present'. Beim Schreiben kann die Fehlermeldung ignoriert werden (sie entsteht auch beim Anlegen des ersten Datensatzes). Dadurch wird nur signalisiert, daß der angesprochene Datensatz noch nicht exisiert und neu angelegt wird.

Beim Lesezugriff dürfen Sie diese Fehlermeldung allerdings nicht übergehen. Die Floppy meldet so, daß versucht wurde, auf einen Datensatz zuzugreifen, der gar nicht vorhanden ist.

1.5.7 SUCHEN EINES RECORDS

Das Suchen von Daten ist ein leidiges Problem. Man könnte leicht ein ganzes Buch mit diesem Thema füllen. Das liegt zum einen daran, daß es hier keine optimale Lösung gibt. So haben sich die Experten 1001 Verfahren ausgedacht, Daten zu ordnen, zu suchen und zu verwalten. Es gibt viele mehr oder weniger praktikable Verwaltungsmethoden. Eine Universalmethode ist allerdings noch nicht erfunden worden (Sie könnten sich dabei also noch ein paar Lorbeeren verdienen). Aus diesem Grund kann die Problematik hier nur angerissen werden. Wer sich damit intensiver beschäftigen möchte, findet im Literaturverzeichnis entsprechende Hinweise.

Das Hauptproblem bei der relativen Datei besteht darin, daß jeder Datensatz nur über die Datensatznummer angesprochen werden kann. Bei unserem Beispiel, der Geburtstagsdatei, ist diese Methode allerding wenig sinnvoll. Schließlich sucht man entweder alle Personen, die an einem bestimmten Datum Geburtstag haben, oder man möchte die Daten einer ganz bestimmten Person finden. Sie können jetzt natürlich damit beginnen, Ihren Verwandten Nummern zuzuweisen. Bei 007 klappt das vielleicht noch aufgrund der Berühmtheit dieser Zahl. Aber hat jetzt Tante Klara die Nummer 102 oder 93? Oder wollen Sie gar, wenn Sie an Tante Klara denken, in Zukunft nur noch von Person Nr. 1652 reden. Dabei müßten Sie nur aufpassen, daß Sie bei der Geburtstagsparty von 672 nicht vergessen, 7362 nach dem Wohlbefinden von 373 und deren Tochter 6292 zu fragen...

Das Problem ist klar. Die Nummerierung der Datensätze ist für den Computer, dem Meister im Zahlenjonglieren, sehr praktisch. Doch der Mensch kann damit nicht arbeiten. Vielleicht können Sie sich jetzt aber vorstellen, warum Versandunternehmen usw. so scharf auf Ihre Kundennummer o.ä. sind.

Nun zur Lösung des Problems. Unser erster Gedanke war gar nicht so dumm. Durch die Datensatznummer wird jedem Namen eine Zahl zugeordnet. Dies müßten wir jetzt rückgängig machen. Dies bedeutet, daß man braucht eine Liste mit den Namen bzw. dem Geburtsdatum benöigt, in der die entsprechende Daten-

satznummer zugeordnet ist. Die relative Datei ist im Grunde genommen das gleiche, nur funktioniert sie umgekehrt.

Um beim Suchen einigermaßen effektiv zu arbeiten, sollten die Namen geordnet sein. Man erstellt also ein Programm, das alle Namen und die dazugehörigen Datensatznummern aus der relativen Datei einliest und alphabetisch sortiert. Diese Daten kann man beispielsweise als sequentielle Datei abspeichern. Die Daten werden dann jedesmal in den Computerspeicher einlesen. Dort ist es möglich, den gesuchten Namen schnell zu finden. Damit kennt man auch die Datensatznummer und kann die restlichen Informationen von Diskette lesen.

Betrachten wir die Sache etwas differenzierter. Im Grunde genommen kommt es doch nur darauf an, daß man die Datensätze sortiert. Nur so verläuft das Suchen schnell genug. Eventuell gibt esja auch raffiniertere Suchmethoden. In der relativen Datei sind die Daten ungeordnet abgespeichert. Die Idee der sortierten Namen und zugeordneten Satznummern ist eigentlich nicht schlecht. Doch wir haben dadurch noch eine zusätzliche Datei, verbrauchen viel mehr Speicherplatz, benötigen auch noch Speicher im Computer usw. Außerdem ist die Zugriffszeit vielleicht recht schnell, nur wird für zusätzlichen Arbeiten (Sortieren, sortierte Liste einlesen,...) sehr viel Zeit verbrauchen.

Genau betrachtet geht es ja nur darum, zu wissen, in welcher Reihenfolge die Datensätze aufgerufen werden müssen, damit sie nach einem bestimmten Kriterium geordnet sind. Dieses Kriterium kann der Name, das Datum oder ein anderes beliebiges Datenfeld der relativen Datei sein. Man nennt dies den Schlüsselbegriff (engl. Key). So gesehen ist es gar nicht erforderlich eine geordnete Liste mit Namen und Satznummer anzufertigen. Es würde ausreichen die Datensatznummern zu sortieren. Die erste Nummer entspricht dann dem alphabetisch ersten Namen, die zweite Nummer dem alphabetisch zweiten Namen usw. Es handelt sich bei dieser Methode immer noch um eine Datei, die man zusätzlich führt - praktischerweise meistens als sequentielle Datei. Doch der Speicherbedarf, sowohl auf Diskette als auch im Computer, wurde wesentlich kleiner, da man den Platz für die Namen spart.

Das ganze hat aber einen Nachteil. Die Forderung bei den relativen Dateien bestand darin, daß man nichts mehr im Speicher behalten muß. Hier ist es aber erforderlich vor jeder Dateianwendung die Daten der Schlüsseldatei einzulesen.

So wie die Schlüsseldatei jetzt geordnet ist, kann man eine Position im Alphabet (z.B. den 5. Namen) anwählen und erhält dann die Datensatznummer des Namens. Genau genommen wollen wir aber nur wissen, welcher Datensatz der erste ist, welcher der nächste ist usw. Die Schlüsseldatei müßte nur angeben, welcher Datensatz der nächste im Alphabet ist.

Man spricht in diesem Fall von einem sogenannten Zeiger oder Index. Wir bringen in jedem Datensatz einfach eine Nummer unter. Diese ist die Datensatznummer des alphabetisch nächsten Datensatzes. Auf diese Art und Weise sind die einzelnen Datensätze miteinander verkettet. Das einzige, was man sich noch merken muß, ist die Datensatznummer, bei der diese Kette beginnt. Und was ist mit dem Ende? Das ist Ihrer Fantasie überlassen. Entweder können Sie hier wieder die Nummer des alphabetisch ersten Datensatzes angeben, oder aber eine 0 zuweisen. Den Datensatz 0 gibt es nicht, also kann Ihr Programm immer das Ende erkennen.

Natürlich funktioniert das alles auch umgekehrt. Sie können eine weitere Kette erstellen, in der die Namen umgekehrt sortiert sind. Dadurch ergibt sich die Möglichkeit, sowohl den nächsten als auch den vorhergehende Datensatz aufzurufen - komfortabler geht es nicht mehr.

Eines können wir uns natürlich nicht sparen: die Datensatznummern, bei der diese Verkettungen beginnen, müssen vermerkt werden. Selbstverständlich könnten Sie dazu eine sequentielle Datei einrichten. Praktischer ist es allerdings, eine Datei erst ab dem 2. oder 3. Datensatz beginnen zu lassen. So kann man 1. oder 2. Satz für derartige Informationen verwenden. Diese Organisationsweise ist allerdings eine Angelegenheit Ihres Programms.

Soll die relative Datei verkettet werden, dann müssen Sie für die Verkettungszeiger in jedem Datensatz entsprechend Platz einplanen. Natürlich könnte man die Datei auch noch nachträglich

verketten. Dazu legt man die Zeiger einfach in einer parallel organisierten relativen Datei ab. Dies erfordert kaum mehr Speicherplatz. Doch kann bei der C-1570/71 immer nur eine relative Datei geöffnet sein. Sie müssten also immer die Hauptdatei schließen, dann die Schlüsseldatei öffnen, die Schlüsseldaten auslesen, dann die Schlüsseldatei wieder schließen und die Hauptdatei öffnen... - das ist nicht nur programmier- sondern vor allem auch zeitaufwendig.

Wir haben jetzt also schöne Verkettungen erstellt. Was aber passiert, wenn man einen neuen Eintrag in die Datei einfügen möchte? Das ist kein Problem. Der neue Datensatz wird einfach ans Ende der Datei angehängt, den die physikalische Lage in der Datei spielt keine Rolle. Nun muß man nur noch den neuen Datensatz korrekt in die Verkettungen einbauen. Dazu sucht man einfach die Stelle, nach der der Datensatz in eine Kette eingebaut werden müßte. Dann liest man hier die Nummer des nächsten Datensatzes. Diese trägt man jetzt im neuen Datensatz ein. Der Datensatz, nach dem eingefügt werden soll, erhält den Zeigerwert des neuen Datensatzes.

Mit der Zahl der Verkettungen, das heißt der Zahl der Schlüsselbegriffe, wächst also der Aufwand beträchtlich. Deshalb muß man sehr sorgfältig abwägen, wie viele und welche Schlüssel die Datei haben soll.

1.5.8 ANWENDUNG DER RELATIVEN DATEI

Nach all diesen theoretischen Überlegungen soll nun wieder etwas Praxis folgen. Das Thema war eine Geburtstagsdatei. Die Länge der Schlüssel hatten wir ja bereits in 1.5.1 festgelegt.

Dabei taucht auch schon das nächste Problem auf. Wir benötigen auch noch Platz für die Verkettungszeiger. Doch wie groß sind diese? Eine Zahlenvariable wird immer als String abgelegt. Das heißt sie kann zwischen einem und 5 Zeichen (bei max. 65535 Datensätzen) lang sein. Es muß also für 4-5 Zeichen Platz sein vorhanden sein, obwohl oft weniger benötigt wird.

Hier hilft uns der Binärmathematiker. Jede Zahl zwischen 0 und 65535 kann man in einen 2-Byte-Binärwert umrechnen. Wir brauchen also für unsere Verkettungszeiger pro Datensatz immer exakt zwei Bytes. Eine praktische Sache. Die Umrechnung ist auch nicht sehr problematisch:

```
a = verkettungszeiger (0..65535)
PRINT#1, CHR$(a and255) CHR$(a/256);
```

Den Wert zurückzuwandeln funktioniert folgendermaßen:

```
GET#1,a$ : GET#1,b$
a = ASC(a$+CHR$(0)) + ASC(b$+CHR$(0)) * 256
```

Nehmen wir an, der Name und das Geburtsdatum sollen als Schlüsel verwendet werden. Sicherlich werden Sie sich schon gefragt haben, wie man die Datei jetzt sortiert und alle Verkettungen anbringt. Wenn Sie die Verkettungen nachträglich installieren möchten, ist das nicht ganz einfach. Deshalb sollte man schon von Anfang an die Schlüssel festlegen. Dann geht man nämlich genauso vor wie beim Anfügen, das wir im vorigen Kapitel besprochen haben.

Wenden wir uns jetzt den einzelnen Datensätzen zu. Die Datensatzlänge muß fest definiert werden, daran ist nicht zu rütteln. Das bedeutet aber noch nicht, daß man die Länge der einzelnen Datenfelder genauso festlegen muß. Natürlich ist dies einfacher.

Diese Überlegung hängt auch eng mit den verwendeten Lesebefehlen zusammen. Möchten Sie INPUT# oder GET# verwenden? Beim 'Input#'-Befehl muß das einzelne Datenfeld mit CR abgeschlossen sein, es wird also zusätzlicher Platz verbraucht. Dies könnte sich aber durchaus lohnen, da Sie mit dem 'Input#'-Befehl sehr einfach variable Datenfeldlängen handhaben lassen.

Dabei wird jedes Datenfeld immer mit CR abgeschlossen. Schon beim Abspeichern müssen Sie allerdings darauf achten, daß die Feldlänge 88 bzw. 160 Zeichen nicht überschreitet. Hat das Feld keinen Inhalt, dann speichert man nur ein CR ab. Der Vorteil besteht darin, daß man kaum noch das Problem hat, daß zum Beispiel ein Name nicht in ein Datenfeld paßt. Es besteht nur

noch die Gefahr, daß die Länge aller Datenfelder, inclusive der CR-Zeichen, die Sie auf gar keinen Fall vergessen dürfen, größer als die Datensatzlänge ist. Daher sollten Sie in Ihrem Programm auf jeden Fall vorher die Gesamtlänge berechnen und gegebenenfalls den Bediener dazu auffordern, Eingaben zu kürzen. Der Nachteil der variablen Datenfelder besteht darin, daß man immer alle Felder vom Anfang des Datensatzes an einlesen muß, um ein bestimmtes Datenfeld zu erreichen. Bei den festen Feldlängen kann man den Zeiger auf die aktuelle Zeichenposition mit dem Record#-Befehl direkt auf das Datenfeld richten.

Beispiele zu diesem Thema sind in Ihrem Bedienungshandbuch enthalten. Es ist auch nicht Sinn dieses Kapitels, Ihnen eine Lösung als das Nonplusultra anzubieten - denn dieses gibt es nicht. Vielmehr wollte ich Ihnen einige Anregungen und Tips zur eigenen Programmierung geben.

KAPITEL 2

PROGRAMMIERUNG FÜR FORTGESCHRITTENE

2.1 DIE DIREKTZUGRIFFSBEFEHLE

2.1.1 DER DIREKTZUGRIFF AUF EINZELNE SEKTOREN

Vielleicht haben Sie sich, aufgrund der Datenverwaltungskapitel einmal in die weiterführende Literatur angeschaut. Dann kennen Sie jetzt einige neue, hervorragende Verwaltungsmethoden, die Sie gerne in die Realität umsetzen möchten. Doch die C-1570/71 kennt diese Verfahren nicht. Wir können also Verwaltung, Organisation usw. nicht mehr der Floppy überlassen, sondern müssen selbst aktiv werden. Dazu benötigen wir Befehle, um die einzelnen Sektoren der Diskette direkt anzusprechen. Somit können wir unsere eigene Datenverwaltung aufbauen. Um genau diese Möglichkeiten geht es in diesem Kapitel. Dazu werden Sie ab 2.1.2 zuerst einmal die erforderlichen Befehle kennenlernen. In den Kapiteln 2.2 und 2.3 geht es darum, wie die normalen Daten (Programme, Dateien) auf der Diskette organisiert sind. Zum einen können Sie sich dort vielleicht für Ihre eigenen Anwendungen etwas abschauen, zum anderen sind diese Kenntnisse unbedingt erforderlich, damit Sie Ihre eigene Dateiform in die normale Diskettenverwaltung einbinden können. Für unsere ersten Experimente ist es deshalb ratsam, eine neu formatierte, leere Diskette zu verwenden. So gehen Sie keine Gefahr ein, wertvolle Daten durcheinanderzubringen oder zu zerstören.

Wie Sie aus den ersten Kapiteln wissen (1.1.2) ist die Diskette in Spuren und Sektoren organisiert. Da der Umfang der äußeren Spur größer, als der der Inneren ist, passen natürlich auf diese Spur auch mehr Sektoren. Die Spuren werden dabei von außen nach innen durchnumeriert. So liegt also ganz außen Spur 1, die die meisten Sektoren enthält. Die innerste Spur hat die Nummer 35. Theoretisch können mit der C-1570/71 bis zu 40 Spuren beschrieben werden. Beim COMMODORE-Format werden diese letzten fünf Spuren aber nicht verwendet.

Eine auf diese Weise formatierte Diskette hat eine Kapazität von 170 KBytes. Dies wäre zum Beispiel bei der C-1570 der Fall. Die C-1571 hat zwei Schreib-/Leseköpfe und kann deshalb beide Seiten der Diskette beschreiben. Folglich ist die Kapazität auch

doppelt so groß. Doch bei beidseitigen Disketten gäbe es ja plötzlich zwei Spuren mit der Nummer 1, jeweils eine auf jeder Seite. Woher weiß die Floppy, wenn man einen bestimmten Sektor anspricht, auf welcher Seite dieser liegt? Aus diesem Grund benennt man die erste Spur auf der zweiten Diskettenseite einfach mit Spur 36.

Es ergeben sich also folgende Sektorenanzahlen:

Spur	Sektornummer
1 - 17	0 - 20
18 - 24	0 - 18
25 - 30	0 - 17
31 - 35	0 - 16
36 - 52	0 - 20
53 - 59	0 - 18
60 - 65	0 - 17
66 - 70	0 - 16

Wenn man die Floppy anweisen kann, einen Sektor von Diskette zu lesen, ergibt sich die Frage, wohin damit ? Da sich die Diskette mit 300 U/min dreht, werden die einzelnen Zeichen mit einer Geschwindigkeit von bis zu über 60000 Bytes/sec gelesen. Ihr BASIC-Programm ist auf keinen Fall in der Lage derartige Datenraten zu verarbeiten. Der Sektor muß also irgendwo zwischengespeichert werden, damit man ihn mit normalen Befehlen bearbeiten kann. Beim Schreiben eines Sektors ist das natürlich nicht anders.

Die Floppy hat deshalb 4 Pufferspeicher, die jeweils genau 256 Bytes, also die Länge eines Sektors, umfassen. Diese Speicher werden auch benötigt, wenn Sie Programme laden, mit Dateien arbeiten usw. Die folgende Tabelle gibt Aufschluß darüber, welche Dateiform wie viele Puffer benötigt:

relative Datei	3 Puffer
Programm laden/speichern	1-2 Puffer
sequentielle Datei	1-2 Puffer
Directory anfordern	1 Puffer
Direktzugriff	1 Puffer

Jetzt wissen Sie auch, warum man zwei relativen Dateien nicht gleichzeitig eröffnen kann - der Pufferspeicher reicht nicht aus. Im Zeitalter der billigen, riesigen Speichermengen ist es schon seltsam daß die Floppy nur über einen Pufferspeicher von 1.25 KByte verfügt. Aber wir müssen uns damit abfinden und versuchen, das Beste daraus zu machen.

Um auf beliebige Sektoren zugreifen zu können, müssen wir uns einen Puffer reservieren. Man nennt dies auch Direktzugriffsmethode. Zuerst geht es darum, einen Datenkanal für den Direktzugriff einzurichten. Dies geschieht mit folgendem Befehl:

```
OPEN x,y,z,"#a"
```

Theoretisch können Sie auch den BASIC-7.0-Befehl 'Dopen' verwenden. Da man später aber die Sekundäradresse z angeben muß, ist es praktischer, den BASIC-3.0-Befehl einzusetzen. Bei BASIC 7.0 wird diese nämlich vom BASIC selbständig gewählt. Die Parameter x und y geben die Kanalnummer und Geräteadresse an.

Als Dateiname wird ein '#' angegeben. Dies signalisiert der Floppy, daß ein Direktzugriffskanal eingerichtet werden soll. Dabei ordnet die C-1570/71 dem Kanal einen Puffer zu. Dessen Nummer (0..3) kann in 'a' angegeben werden. Normalerweise sollten Sie diese Angabe weglassen. Die Floppy wählt dann automatisch einen freien Puffer. Sie könnten sonst eventuell einen Puffer anwählen, der bereits für andere Zwecke benutzt wird.

Sind alle oder der gewünschte Puffer belegt, dann meldet sich die Floppy mit '70 No Channel'. Prüfen Sie also immer nach dem Eröffnen des Direktzugriffskanals die Fehlervariable 'ds'.

2.1.2 BLOCK-READ UND BLOCK-WRITE

Wie schon angekündigt gibt es spezielle Befehle um einen bestimmten Sektor in den Floppypuffer einzulesen oder vom Puffer auf die Diskette zu schreiben. Diese Befehle werden über den Befehlskanal, also Kanal 15 übermittelt. Daher sollten Sie bei allen Direktzugriffen zuerst einmal den Befehlskanal (siehe auch 1.3.1) öffnen. Die einzelnen Sektorbefehle haben immer den gleichen Aufbau:

"aaa:k l t s"

Die einzelnen Parameter haben folgende Bedeutung:

aaa	Befehlswort
k	Kanalnummer (Sekundäradresse)
l	Laufwerksnummer (0/1)
t	Spurnummer (0..35/70)
s	Sektornummer

Die Kanalnummer k ist der Parameter z beim Eröffen des Direktzugriffskanals aus dem vorherigen Kapitel. Jetzt ist Ihnen sicherlich klar, warum wir diesen Parameter unbedingt wissen wollten. Die Laufwerksnummer l hat bei der C-1570/71 keine Funktion, da es sich um ein Einzellaufwerk handelt. Trotzdem dürfen Sie diese Angabe nicht weglassen. Sie lautet immer 0. Zuletzt folgen die Daten des gewünschten Sektors, also Spur und Sektornummer. Die einzelnen Parameter werden im Befehlsstring durch Leerzeichen oder Kommata getrennt. Das Befehlswort kann durch ein Leerzeichen oder einen Doppelpunkt von den Parametern getrennt werden.

Nun zu diesem Befehlswort. Damit wird die genaue Funktion (Lesen/Schreiben) ausgewählt. Merkwürdigerweise gibt es für ein und dieselbe Sache gleich mehrere Befehle:

Lesen	Schreiben
b-r u1 ua	b-w u2 ub

Die Befehle u1/ua bzw. u2/ub sind völlig identisch. Damit wird ein angegebener Sektor in den Puffer gelesen oder vom Puffer auf Diskette geschrieben. Es kann dabei immer auf alle Bytes des Sektors zugegriffen werden. Die Befehle 'b-r' bzw. 'b-r' führen nichts anderes aus. Nur ist es nicht mehr möglich, alle Bytes des Sektors zu lesen oder zu schreiben. Dies ist auf einen Fehler im Betriebssystem zurückzuführen. Das Bedienungshandbuch zur C-1570/71 beschreibt diese Sonderfunktion als eine ganz hervorragende Sache. Sie läßt sich aber nur sinnvoll einsetzen, wenn mit dem letzten Sektor eines Programms oder einer sequentiellen Datei gearbeitet werden soll. In der Praxis bedeutet dies, daß man 'b-r' und 'b-w' getrost vergessen kann. Deshalb soll auf diese Befehle auch nicht weiter eingegangen werden.

Jetzt ergibt sich natürlich die Frage, wie man bestimmte Bytes aus dem Puffer in den Rechner überträgt? Dies funktioniert wie bei den bisherigen Datenkanälen, mit dem 'Get#'- oder dem 'Input#'-Befehl, wobei man allerdings meistens den 'Get#'-Befehl verwenden wird. Input# ist selbstverständlich auch möglich, doch muß hier gewährleistet sein, daß nach maximal 87/159 Zeichen (64er/128er-Modus) ein CR folgt.

An welcher Stelle im Puffer die Bytes für einen Lese-/Schreibbefehl geholt bzw. geschrieben werden sollen, wird durch den Pufferzeiger festgelegt. Dieser ist nach der U1/U2-Operation auf den Beginn des Puffers gesetzt. Möchte man aber auf einen

beliebigen Teil des Puffers (Sektors) zugreifen, dann kann man dazu den Block-Pointer-Befehl verwenden. Dieser lautet:

"b-p k b"

Die Angabe k ist die Kanalnummer (Sekundäradresse), die Sie beim Eröffnen des Direktzugriffskanals angegeben haben. Mit b kann die Position des Pufferzeigers bestimmt werden. Die Stelle b ist dann die Position, die vom nächsten Schreib- oder Lesebefehl betroffen ist. Da ein Sektor und somit auch ein Puffer 256 Bytes groß ist, darf b nur Werte zwischen 0 und 255 annehmen.

In Programmen wird man für die Parameter, wie Spur und Sektor, meistens Variablen verwenden wollen. Das ist kein Problem. Gibt man eine Variable über den 'Print#'-Befehl aus, so wird das zur Trennung der Parameter erforderliche Leerzeichen immer mitgeliefert. Dies ist nämlich das Vorzeichen des Variablenwerts, das bei positiven Werten als Leerzeichen gedruckt wird.

Nun ein Beispiel, wie Sie die U1/U2-Befehle einsetzen können:

```
OPEN 1,8,15                 Befehlskanal öffnen
OPEN 2,8,2,"#"              Zugriffskanal öffnen
IF ds<>0 THEN PRINT DS$     Puffer frei?
INPUT "spur ";t             Spur eingeben
INPUT "sektor ";s           Sektor wählen
PRINT#1,"u1:2 0";t;s        Sektor in Puffer lesen
IF ds<>0 THEN PRINT DS$     Sektor richtig gelesen?
```

Jetzt dürfen Sie an den Daten Manipulationen vornehmen:

```
PRINT#1,"b-p";2;10          Pufferzeiger setzen
PRINT#2,"neue daten"        in Puffer schreiben
PRINT#1,"u2:2 0";t;s        Sektor zurückschreiben
IF ds<>0 THEN PRINT DS$     Sektor geschrieben?
CLOSE 1                     Kanal 1 und 2 schließen
```

Close 1 reicht aus, da beim Schließen des Befehlskanals alle anderen Kanäle auch geschlossen werden.

2.1.3 BLOCK-ALLOCATE UND BLOCK-FREE

Immer wieder haben Sie davon gehört, daß die Floppy auf der Diskette auch vermerkt, welche Sektoren belegt und welche noch frei sind. Ist ein Sektor als belegt gekennzeichnet, kann er von normalen Datei- und Programmdaten nicht beschrieben werden. Die Sektorbefehle wiederum werden dadurch nicht behindert.

Deshalb gibt es spezielle Befehle, um einzelne Sektoren zu belegen oder freizugeben. Diese lauten:

"b-f l t s" zum Freigeben
"b-a l t s" zum Belegen

Die Angabe l ist die Laufwerksnummer (also immer 0), t und s sind Spur- und Sektornummer des gewünschten Blocks - im Grunde genommen eine einfache Sache, hätte nicht wieder einmal im Diskettenbetriebssystem der Fehlerteufel zugeschlagen. Gibt man beim 'b-a'-Befehl Sektornummern über 15 an, dann wird nicht nur der angewählte Sektor, sondern gleich die ganze Spur belegt. Belegt man einen Sektor, der schon belegt ist, dann soll der Befehl normalerweise den nächsten freien Sektor suchen. Findet er aber auf der Spur keinen, dann probiert er es bei der nächst höheren Spur. Diese wird aber vollständig belegt. Auch der 'b-f' Befehl funktioniert nur mit Sektornummern bis 15. Eine höhere Sektornummer "interessiert" ihn überhaupt nicht - der Befehl überhaupt nicht ausgeführt. Fazit: entweder Sie beschränken sich bei Ihren Anwendungen auf die Sektoren 0 bis 15 oder unterlassen es besser ganz. Wenn Sie für Ihre im Direktzugriff verwalteten Daten jeweils eigene Disketten verwenden, spielt die Blockverwaltung sowieso keine Rolle.

Zum Schluß noch eine Bemerkung. Sollen 'b-a' und 'b-r' überhaupt funktionieren, ist es erforderlich die Diskette vorher mit "i" (über den Befehlskanal) zu initialisieren.

2.2 DER AUFBAU DER DISKETTE

2.2.1 DAS DIRECTORY

Sicherlich haben Sie sich schon gefragt, wie das Directory funktioniert. Schließlich muß es auch irgendwo auf der Diskette stehen. Damit die Arbeit mit den Directory-Einträgen schnell genug abläuft, sind diese nicht wild verstreut, sondern haben eine eigene Spur. Bei COMMODORE-Disketten ist das die Spur Nummer 18. Diese Spur kann von anderen Daten nicht belegt werden. Auch sollten Ihre Direktzugriffsdateien nicht gerade diese Spur belegen. Nun ist es verständlich, warum von insgesamt 683 bzw. 1366 Sektoren nur 664 bzw. 1328 zur Datenspeicherung zur Verfügung stehen.

Das Directory, also die Dateieinträge, belegen dabei die Sektoren 1-18 der Spur 18 auf der ersten Diskettenseite. Dabei werden die Sektoren nicht in der numerischen Folge benutzt, sondern mit einem Versatz von 3 Sektoren. Das heißt : auf Sektor 1 folgen die nächsten Daten in Sektor 4 usw. Ist das Ende der Spur erreicht werden die anderen Sektoren (jetzt zum Beispiel ab Sektornummer 2) in der gleichen Art und Weise belegt.

Jeder Sektor kann dabei maximal acht Einträge aufnehmen. Man kann also auf einer Diskette bis zu 144 Programm- oder Dateititel abspeichern.

Im folgenden wollen wir jetzt den genaueren Aufbau eines Directory-Sektors kennenlernen. Die ersten beiden Bytes sind die Verkettungszeiger. Diese zeigen an, welche Spur- und Sektornummer der nächste Directory-Sektor hat. Folgt kein weiterer Sektor, dann hat das erste Byte, das die Spurnummer repräsentiert, den Wert 0. Das zweite Byte gibt immer an, wie viele Bytes des letzten Sektors noch dazugehören.

Danach folgt der 1. Eintrag des Sektors, dann zwei Leer-Bytes, anschließend der 2. Eintrag und so weiter. Das entscheidende ist also der Aufbau eines einzelnen Dateieintrags, der jetzt näher behandelt werden soll.

Byte	Bedeutung
0	Dateityp Bit 0-3: 0 DEL Eintrag gelöscht 1 SEQ sequentielle Datei 2 PRG Programm 3 USR User-Datei 4 REL relative Datei Bit 6: 1= kein Schreibzugriff erlaubt Bit 7: 1= Eintrag ordnungsgemäß geschlossen
1/2	Spur- und Sektornummer des ersten Datenblocks des Eintrags.
3-18	Dateiname des Eintrags (max. 16 Zeichen). Der Rest wird mit 'Shift+Space' (ASCII-Wert 160) auf gefüllt.
19/20	Spur- und Sektornummer des ersten Side-Sektor-Blocks. Wird nur bei relativen Dateien benutzt.
21	Länge eines Datensatzes. Wird nur bei relativen Dateien benutzt.
22-25	unbenutzte Bytes
26/27	Zwischenspeicher für Spur- und Sektornummer des ersten Datenblocks der neuen Datei, wenn mit der '@'-Funktion überschrieben wird.
28/29	Low- und High-Byte der Zahl der benutzen Blöcke der Datei. Die Zahl ist in Binärform abgelegt.

Nun wollen wir die wichtigsten Punkte des Dateieintrags besprechen. Das ist hauptsächlich das Dateityp-Byte. In diesem ist vermerkt, um welche Dateiform es sich handelt. Die Abkürzungen aus dem Directory sind Ihnen sicher schon bekannt. Was aber ist 'DEL'. Dies zeigt an, daß der Eintrag gelöscht ist. Derartige Einträge werden normalerweise im Directory nicht aufgelistet. Die

Floppy übergeht bei der Anzeige des Directorys nämlich alle Einträge, deren Dateityp 0 ist. Wenn Sie jetzt Bit 7 auf 1 setzen, würde im Directory tatsächlich ein Eintrag mit der Dateitypkennung 'DEL' aufgelistet, da jetzt der Dateityp nicht mehr 0 ist.

Das Bit 7 zeigt an, ob die Datei richtig geschlossen wurde. Wird auf der Diskette eine neue Datei angelegt oder ein neues Programm abgespeichert, so wird zuerst der Directory-Eintrag erstellt - auch um zu prüfen, ob nicht bereits ein gleichnamiger Eintrag besteht. Dabei wird auch der Dateityp schon vermerkt, lediglich das Bit 7 wird noch nicht gesetzt. Sind die Daten abgespeichert, wird die Datei geschlossen. Dabei wird in die Bytes 28/29 die Zahl der belegten Blöcke eingetragen und Bit 7 des Dateitys auf 1 gesetzt. Dadurch ist der Eintrag gültig. Ist Bit 7 nicht gesetzt, dann wird das im Directory-Ausdruck mit einem Stern vor dem Dateityp markiert. Auf diese Weise markierte Dateien sind beispielsweise solche, bei denen Sie den 'Close'-Befehl vergessen haben. Mit dem Setzen des Bits 7 ist der Fehler allerdings noch nicht behoben. Sie sollten auf jeden Fall noch einen 'Collect'-Befehl durchführen, um die anderen Diskettenverzeichnisse in Ordnung zu bringen. Dadurch ist auch garantiert, daß die Datei wieder voll benutzbar ist.

Eine Sonderfunktion hat Bit 6. Ist es gesetzt, dann kann man auf den Eintrag keine Schreiboperationen vornehmen. Das heißt der 'Scratch'- oder der 'Rename'-Befehl wirken nicht mehr. Die Datei (das Programm) kann nur noch gelesen werden. Leider gibt es keinen Befehl, mit dem dieses Bit gesetzt oder gelöscht werden kann. Das können Sie nur "von Hand" machen, indem Sie den Diskmonitor aus Kapitel 6.1 verwenden.

2.2.2 DIE BLOCKBELEGUNGSTABELLE - BAM

In den vorherigen Abschnitten wurde mehrfach erwähnt, daß in einer Tabelle eingetragen ist, welche Blöcke der Diskette noch zum Beschreiben frei sind und welche belegt sind. Diese Tabelle nennt man bei COMMODORE-Disketten BAM, was eine Abkürzung für 'Block-Availability-Map' (engl. = Blockverfügbarkeitstabelle) ist.

Die BAM ist in Sektor 0 der Directory-Spur (18) untergebracht. Außerdem enthält dieser Sektor den Diskettennamen, den Sie beim Formatieren angegeben haben.

Bevor wir uns mit dem Gesamtaufbau des Blockverzeichnisses beschäftigen, betrachten wir einmal, wie die Sektorbelegung einer Spur gespeichert ist.

Blöcke	Sektor 0-7	Sektor 8-15	Sektor 16-23
$12	%11111111	%11111111	%11000000

1= Sektor frei 0= Sektor belegt

Für jede Spur sind in der BAM vier Bytes vorhanden. Das erste Byte ist ein Binärwert, der die Zahl der freien Blöcke der Spur angibt. Die darauffolgenden drei Bytes enthalten ein Bitmuster, bei dem jedes Bit einem Sektor entspricht (sofern die Sektornummer vorhanden ist). Hat das Bit eines Sektors den Wert 1, dann bedeutet dies, daß der Sektor noch frei ist. Setzt man das Bit auf 0, dann kann der Sektor vom Floppybetriebssystem nicht mehr beschrieben werden.

Die Blockangabe im ersten Byte ist einfach eine Arbeitserleichterung für das C-1570/71-Betriebssystem. So muß es nicht jedesmal alle auf 1 gesetzten Bits zählen, was eine zeitaufwendige Sache ist.

Sie können Ihre BAM sogar mit Hilfe eines Diskettenmonitors so manipulieren, daß die Zahl der freien Blöcke der Spur nichts mehr mit der Realität zu tun hat. Sie können problemlos eintragen, daß die Spur 255 freie Blöcke hat, was natürlich nicht möglich ist. Da diese Blockangabe auch beim Errechnen der Gesamtzahl der freien Blöcke, die Sie im Directory immer unter 'Blocks free' finden, verwendet wird, sind astronomische Zahlen von bis zu über 17000 freien Blöcken möglich - die Sie in Wirklichkeit natürlich gar nicht haben.

Das Betriebssystem der Floppy macht diese Spielereien allerdings nicht lange mit. Bei allen Veränderungen in der BAM wird

geprüft, ob die Blockangaben der Spuren noch mit ihren jeweiligen Bitmustern übereinstimmen. Wird hier eine Abweichung gefunden, so meldet sich die C-1570/71 mit '71 Dir Error'.

Die gesamte BAM befindet sich in Sektor 0 der Directory-Spur. Dieser Sektor hat folgenden Aufbau:

Byte	Bedeutung
0/1	Spur- und Sektornummer des ersten Directory-Sektors, normalerweise Spur 18 Sektor 1.
2	Formatkennzeichen, bei 4040/1541/1570/1571 immer 'A' (ASCII-Wert 65). Bei falschem Formatkennzeichen ist die Diskette schreibgeschützt.
3	Bit 7: 0= einseitige 1541/1570-Diskette 1= zweiseitige 1571-Diskette
4-143	BAM-Verzeichnis für Diskettenseite 1. Jede Spur wird durch vier Bytes repräsentiert. Das Verzeichnis beginnt mit Spur 1.
144-159	Diskettenname, der beim Formatieren angegeben wurde. Bis zu 16 Zeichen. Der Rest wird mit 'Shift + Space' (ASCII-Wert 160) aufgefüllt.
160/161	zwei 'Shift + Space' (ASCII-Wert 160)
162/163	ID-Zeichen der Diskette
164	Dies soll die Versionsnummer des Betriebssystems sein. Das Zeichen lautet aber immer noch '2', obwohl die 1570/71 schon mit DOS 3.0 arbeitet.
165	Formatkennzeichen von Byte 3
167-170	drei 'Shift + Space' (ASCII-Wert 160)
171-220	49 Nullwerte
221-255	Zahl der verfügbaren Sektoren pro Spur auf der Rückseite der Diskette. Dabei handelt es sich um die Blockangaben aus dem BAM-Verzeichnis für die Rückseite. In Byte 221 ist der Wert von Spur 36. Die letzte Angabe, in Byte 255, betrifft die Spur 70.

In Byte 2 ist ein Kennzeichen gespeichert, das den Formattyp angibt. Lautet dieses Zeichen nicht 'A', dann setzt die Floppy voraus, daß ein anderes Format der Directory- und BAM-Verwaltung vorliegt, das sie natürlich nicht durcheinanderbringen möchte. Deshalb sind keine Schreiboperationen zugelassen. Ein derartiger Versuch wird mit der Einschaltmeldung quittiert, womit das Betriebssystem darauf aufmerksam macht, daß es mit diesem Format nichts anfangen kann. Nur die Direktzugriffsbefehle funktionieren noch. Das ist auch einleuchtend. Denn der Sektorschreibbefehl ('u2') kümmert sich um keinen BAM- oder Directory-Eintrag.

Ab Byte 145 ist der Diskettenname abgelegt. Alle Zeichen bis Byte 170 sind aber nur für die Directory-Ausgabe vorhanden und ergeben die Titelzeile. Der Inhalt ist völlig uninteressant. So erscheinen auch die unzähligen ID-Änderungsprogramme, die immer wieder in Fachzeitschriften veröffentlicht werden, in einem ganz anderen Licht. Verändert man hier die ID, erhält man nur eine andere Anzeige in der Titelzeile. An der eigentlichen ID im Header jedes Sektors (siehe 1.1.2) ändert sich auf diese Weise gar nichts.

2.2.3 EIN- ODER ZWEISEITIGE DISKETTEN

Vielleicht haben Sie sich schon gefragt, wo bei zweiseitigen Disketten das Blockverzeichnis (BAM) der zweiten Seite 'versteckt' ist. Der Sektor 0 nimmt schließlich nur die Daten der ersten Seite auf.

Bei zweiseitigen Disketten geht die Directory-Spur auf der Rückseite der Diskette weiter. Die Spur auf der Rückseite hat auch nicht die Nummer 18, sondern die Nummer 53, da die Rückseite mit Spur 36 beginnt.

In Sektor 0 der Spur 53 befindet sich von Byte 0 bis Byte 104 das BAM-Verzeichnis der 2. Seite. Dabei handelt es sich aber nur um die Bitmuster jeder Spur. Das Byte, in dem die Zahl der freien Blöcke der Spur vermerkt ist, ist noch im Sektor 0 auf Spur 18

untergebracht (siehe Tabelle). Ansonsten unterscheidet sich das Verzeichnis nicht von der BAM der 1. Seite.

Die restlichen Sektoren der Spur 53 (Nummer 1 bis 18) sind unbenutzt. Sie können weder für das Directory noch für Dateien o.ä. verwendet werden. Mit der Direktzugriffsmethode können Sie in diesen Sektoren beliebige Daten unterbringen - vielleicht eigene Verwaltungsverzeichnisse oder einen Kopierschutz.

Wird eine zweiseitige C-1571-Diskette in einer C-1541 oder C-1570 verwendet, kann natürlich nur die erste Diskettenseite gelesen werden. Dies ist insofern problematisch, als die Floppy den Zugriff (Programm laden, Daten lesen) mit der Fehlermeldung '67 Illegal Track or Sector' abbricht, wenn die Daten teilweise oder ganz auf der Rückseite der Diskette stehen. Wenn Sie die C-1571 am C-64 oder im C-64-Modus betreiben, verhält sie sich genauso wie ein C-1541-Laufwerk. Es würde also das gleiche Problem auftauchen. Deshalb gibt es Befehle, die in den 1571-Betrieb oder auf die zweite Seite umschalten (siehe Kapitel 3.1.4).

2.2.4 MANIPULATIONEN AN DIRECTORY UND BAM

Das Formatkennzeichen war nur eine der Möglichkeiten, durch Manipulationen an den Verzeichnissen ein bestimmtes Verhalten der C-1570/71 zu provozieren. Und schon blüht der Handel mit diesen Top-Secret-Details. Stolz werden sie unter der Hand weitergereicht und einige Computerheftchen protzen immer wieder damit, ihren Lesern die neusten Tricks und Kniffe zu vermitteln, um das Directory oder die BAM so richtig schön durcheinander zu bringen.

Alle diese Manipulationen kann man im Grunde genommen in zwei Schubladen einordnen. Die eine Gruppe, wie zum Beispiel der Formattrick, dient der Erweiterung der Floppy-Fähigkeiten oder bietet irgendeinen kleinen Nutzen. Die andere Sorte Tricks sind die Basteleien, die das Directory in ein chaotisches Schlachtfeld verwandeln. Sie sollen dazu dienen, daß man beispielsweise bestimmte Programme nicht mehr so einfach laden kann oder der Inhalt der Diskette gar nicht mehr aufgelistet wird o.ä.

Alle diese Methoden können uns als schon erfahrenen Floppy-Programmierer, natürlich nicht schockieren. Im Direktzugriff kann man die BAM- und Directory-Sektoren nämlich auslesen und nachschauen, was dort passiert ist. Besonders der Diskmonitor (siehe 6.1) zeigt alle derartigen Manipulationen an.

Deshalb soll hier auf eine endlose Auflistung dieser Kniffe verzichtet werden. Es gibt aber auch einige sinnvolle Tricks, die uns die Arbeit mit der Floppy erleichtern.

Als erstes wäre da natürlich das Bit 6 des Dateityps zu erwähnen, wodurch man einzelne Einträge vor dem Löschen oder Überschreiben schützen kann (siehe 2.2.1). Ein weiterer beliebter Ort der Manipulation ist der Dateiname. Dabei wird meistens die Möglichkeit ausgenutzt, daß auch bei einem Namen mit weniger als 16 Zeichen alle 16 ausgedruckt werden. Die wurden jedoch durch Leerzeichen aufgefüllt.

Genau das wollen wir ändern. Vielleicht haben Sie auch schon probiert, nach dem Auflisten des Directorys einfach die Blockan-

gabe eines Eintrags mit einem Ladebefehl (Run/Dload/Load) zu überschreiben. Auf diese Weise müssten Sie dann den Dateinamen nicht mehr eintippen. Aber die Sache will nicht so richtig funktionieren. Der Computer reklamiert immer mit 'Syntax Error'. Das ist auch nicht verwunderlich, denn was soll er mit dem Dateitypkürzel anfangen, das noch in der Befehlszeile steht? Wenn Sie dieses mit Leerzeichen überschreiben, funktioniert das Ganze. Doch jetzt ist der Aufwand bald größer als bei normaler Eingabe der Dateinamen.

Genau betrachtet ist es unser Ziel, daß nach dem Dateinamen im Directory noch einige Zeichen ausgegeben werden, die aus der Zeile eine vollwertige BASIC-Befehlszeile machen. Dann muß man wirklich nur noch die Blockangabe überschreiben und schon wird das Programm geladen. Als Endzeichen käme folgendes in Frage:

':'	wenn DLOAD oder RUN verwendet wird.
',8:'	wenn LOAD für BASIC-Programme verwendet wird.
',8,1'	bei LOAD für absolut zu ladende Programme.

Wie Sie in 2.2.1 gelernt haben, muß nun nach dem Dateinamen ein 'Shift+Space' (ASCII-Wert 160) stehen. An dessen Stelle wird nachher im Directory das zweite Anführungszeichen gedruckt. Danach müßte man die obigen Angaben einbauen. Dadurch darf Ihr restlicher Dateiname, je nach eingebauter Angabe, nur noch bis zu 14 Zeichen groß sein. Um diese Manipulation vorzunehmen, brauchen Sie nicht einmal einen hochkomplizierten Diskmonitor aus der Schublade zu kramen. Diese Ergänzungen können einfach beim Abspeichern eingebaut werden:

z.B. DSAVE ("name"+CHR$(160)+",8:")

2.3 DER AUFBAU DER DATEIEN

2.3.1 PROGRAMME, SEQUENTIELLE UND USER-DATEIEN

Nun geht es darum, wie die normalen Programme und Dateien auf der Diskette abgelegt werden. Zuerst werden die Programme und sequentiellen Dateien besprochen. Im nächsten Kapitel wird dann näher auf die relativen Dateien eingegegangen, die schon etwas komplizierter sind.

Die einfachste Form ist immer noch die sequentielle Datei. Die Daten werden nacheinander in die Datei geschrieben. Dabei wandern die Informationen zuerst einmal in den floppyinternen Pufferspeicher. Ist ein Puffer voll, dann wird dessen Inhalt in einen freien Sektor auf die Diskette geschrieben. Dabei muß dieser in der BAM als belegt gekennzeichnet werden. Ist das geschehen, geht man mit den weiteren Daten genauso vor.

Nun besteht natürlich das Problem, daß man wissen muß, welche Sektoren die Datei bilden und in welcher Reihenfolge man sie wieder lesen muß. Dazu befindet sich im Directory-Eintrag ein Zeiger, der die Spur- und Sektornummer des ersten Datensektors enthält (Byte 1/2). Man weiß jetzt zumindest, wo die Datei beginnt. Damit man auch den nächsten und den darauffolgenden Sektor kennt, werden diese verkettet. Dabei ist in jedem Sektor in den ersten beiden Bytes die Spur- und Sektornummer des nächsten Sektors angegeben. Aus diesem Grund kann ein Sektor auch nur 254 Bytes an Daten aufnehmen. Diese Verkettung verläuft so bis zum letzten Datensektor. Dieser hat als Spurnummer des nächsten Sektors eine 0. Daran erkennt die Floppy, daß mit diesem Sektor die Aufzeichnung beendet ist.

Normalerweise werden aber kaum alle Bytes des letzten Sektors noch zur Datei gehören. Deshalb muß man auch wissen, wieviele Bytes noch zur Aufzeichnung gehören. Das wird im zweiten Byte des Sektors (vorher unsere Sektornummer des nächsten Blocks) vermerkt.

Mit dieser Verkettungmethode werden sequentielle Dateien und User-Dateien verwaltet. Was aber ist eine 'User-Datei'? Eigentlich nichts anderes als sequentielle Dateien. Sie werden genauso ange-

sprochen und bedient, wie wir es in 1.4 behandelt haben. Nur muß der Dateityp anstelle von 's' eben 'u' lauten. Man hat so die Möglichkeit, zwischen zwei Bezeichnungen für sequentielle Dateien zu wählen. Auf diese Weise kann man das Directory differenzierter gestalten. Ausserdem gibt es auch einen Floppy-Befehl, der nur mit User-Dateien funktioniert (davon mehr in 2.3.3).

Programme werden nach dem gleichen Verfahren abgespeichert, wie die sequentiellen Dateien. Der einzige Unterschied besteht darin, daß die ersten beiden Bytes einer Programmdatei die Startadresse des Programms bilden (Low-Byte/High-Byte) und keine Daten sind. Für die Floppy spielt das keine Rolle, die Startadresse muß nämlich der Computer beachten. Der C-1570/71 wurde lediglich einprogrammiert, daß sie Programmdateien auf grund dieser störenden Startadresse nicht verkettet. Darüber hinaus hat der Zugriff auf Programme noch zwei eigene Datenkanäle erhalten. Dadurch muß man den Dateityp und die Dateibetriebsart nicht mehr angeben (siehe 1.2.1).

2.3.2 DIE RELATIVE DATEI, DIE SIDE-SEKTOR-BLÖCKE

Die Daten der relativen Datei werden zuerst einmal nicht anders abgelegt, als die einer sequentiellen Datei. Wie sie aber wissen, ist eine relative Datei in Datensätze (Records) organisiert. Man kann auf jeden beliebigen Datensatz zugreifen.

Das Wichtigste ist dabei, daß man die Datensatzlänge vorher fest definieren muß. Deshalb ist es möglich, mit der Datensatznummer und der Datensatzlänge die Zahl der Bytes auszurechnen, die man bis zum gewünschten Datensatz überlesen muß. Würde man die vorherigen Informationen der relativen Datei tatsächlich überlesen, hätte man allerdings nichts gewonnen - das funktioniert auch bei einer sequentiellen Datei.

Die ganze Sache wird aber wesentlich beschleunigt, wenn man die Zahl der Bytes bis zum gewünschten Datensatz durch 254 teilt. Dies ist genau die Anzahl der Bytes, die in jeden Sektor passt. Das heißt, wir können so errechnen, im wievielten Sektor der Verket-

tungsfolge der Datensatz zu finden ist. Der Rest, der bei der Division entsteht, gibt dann an, beim wievielten Byte des Sektors die gewünschten Daten beginnen. Natürlich kann man jetzt die Sektorenverkettung verfolgen, um den richtigen Sektor zu finden. Doch damit wäre man kaum schneller, als bei einer sequentiellen Datei.

Der Clou der relativen Datei besteht darin, daß die Sektorenverkettung in einem speziellen Verzeichnis protokolliert wird. Dieses Verzeichnis besteht aus maximal 6 Sektoren, die man Side-Sektor-Blöcke nennt. Sie sind folgendermaßen aufgebaut:

Byte	Bedeutung
0/1	Spur- und Sektornummer des nächsten Side-Sektor-Blocks.
2	Nummer dieses Side-Sektor-Blocks (0..5)
3	Länge eines Datensatzes der relativen Datei
4/5	Spur- und Sektornummer des ersten Side-Sektors (0)
6/7	Spur- und Sektornummer des zweiten Side-Sektors (1)
8/9	Spur- und Sektornummer des dritten Side-Sektors (2)
10/11	Spur- und Sektornummer des vierten Side-Sektors (3)
12/13	Spur- und Sektornummer des fünften Side-Sektors (4)
14/15	Spur- und Sektornummer des letzten Side-Sektors (5)
16-255	Jeweilige Spur- und Sektornummer der Datenblöcke

Das wichtigste bei den Side-Sektor-Blöcken sind die Bytes 16-255. Hier wird eine Liste der verwendeten Datenblöcke abgelegt. Die Bytes 16/17 sind somit die Spur- und Sektornummer des ersten Datensektors der relativen Datei, die Bytes 18/19 die des zweiten usw. Insgesamt haben so die Spur- und Sektornummern von 120

Datenblöcken in einem Side-Sektor Platz. Um auch eine größere Datei bilden zu können, benutzt man einfach weitere derartige Side-Sektoren.

Nun möchte man aber auch wissen, wo der Sektor liegt, der den gewünschten Datensatz enthält. Dazu muß man nur die vorher errechnete Zahl der Blöcke bis zu dem Sektor, der den Datensatz enthält, durch 120 teilen. Auf diese Weise erfährt man, in welchem Side-Sektor sich die Angaben zum gesuchten Datensektor befinden. Der bei der Division entstandene Rest gibt die Position der Spur- und Sektorangaben im Side-Sektor an.

Aus diese Weise kennen wir jetzt den Sektor, in dem der Datensatz enthalten ist. Weiter kann man mit dem Rest der ersten Division, mit Hilfe derer wir die Zahl der Datenblöcke bis zum richtigen Sektor ausgerechnet hatten, die Position des Datensatzes im Sektor? Eventuell reicht aber ein Teil des Datensatzes bis in den nächsten Sektor hinein. Dies errechnet die Floppy dann noch durch die aktuelle Position und die Datensatzlänge.

Bei der Side-Sektor-Methode braucht man so maximal 3, normalerweise sogar nur 1 bis 2 Sektorzugriffe, bis man den gewünschten Datensatz gefunden hat. Zuerst liest man den ersten Side-Sektor. Wenn man Pech hat, sind die Angaben des errechneten Datensektors nicht in diesem Side-Sektor enthalten. Deshalb werden in jedem Side-Sektor auch immer die Nummern der anderen Side-Sektoren vermerkt (Byte 4-15). So weiß die Floppy nach dem ersten Zugriff auf jeden Fall, in welchem Side-Sektor die richtigen Spur- und Sektorangaben enthalten sind. Danach muß man noch den Side-Sektor lesen. Aus diesem entnimmt man dann die Position des Datensektors. Spätestens beim dritten Zugriff ist der Sektor mit dem richtigen Datensatz gefunden.

Drei Zugriffe stellen aber wirklich den ungünstigsten Fall dar. Normalerweise wird nämlich einer der Side-Sektoren immer im Puffer zwischengespeichert. Somit weiß man sofort, in welchem Side-Sektor-Block sich die gesuchten Angaben zum Datensektor befinden. Es sind also üblicherweise zwei Zugriffe auf die Diskette erforderlich. Wenn man Glück hat und sich schon der richtige Side-Sektor im Puffer befindet oder die Datei noch so klein ist, daß Sie nur einen Side-Sektor umfasst, kann man sogar

direkt den richtigen Datensektor lesen. Dieser Fall kommt gar nicht so selten vor, denn um eine Datei mit mehr als einem Side-Sektor zu erhalten, muß sie erst einmal größer als ca. 30 KByte werden - und das ist gar nicht wenig.

KAPITEL 3
PROGRAMMIERUNG FÜR FREAKS

3.1 PROGRAMME IM DOS-PUFFER

3.1.1 MEMORY-READ UND MEMORY-WRITE

Wie Sie schon im Vorwort gehört haben, wird die C-1570/71 von einem eigenen Mikroprozessorsystem gesteuert. Im weiteren Teil des Floppy-Buches werden wir jetzt intensiver auf diese Interna des Laufwerks eingehen. Es wird auch immer mehr von der Programmierung der Floppy in 6502-Assembler, der Sprache des eingebauten Mikroprozessors, die Rede sein. Die folgenden Teile können Sie aber durchaus auch verstehen, wenn Sie kein Assembler-Profi sind. Was noch nicht ist, kann ja auch noch werden. Deshalb sind im Literaturverzeichnis einige Titel zum Thema Assembler-Programmierung angegeben.

Wie Sie bereits wissen, hat die Floppy einen internen Pufferspeicher. Insgesamt handelt es sich um 2 KByte RAM, die im Bereich von $0000 bis $07FF untergebracht sind. Ein Teil des RAMs wird zu Systemzwecken benötigt, sonst könnte der Mikroprozessor gar nicht funktionieren. Der andere Teil, insgesamt 5*256 Bytes, wird als Pufferspeicher verwendet. In diesen Puffern können aber nicht nur Daten abgelegt werden. Vielmehr ist es auch möglich, darin Programme in 6502-Maschinensprache unterzubringen. Diese können dann ins Betriebssystem der Floppy eingebaut werden.

Nun brauchen wir natürlich wieder einen Befehl, mit dem man das Programm in den Puffer der Floppy schreiben kann. Dazu würde beispielsweise die Direktzugriffsmethode vollständig ausreichen. Man wählt in diesem Fall einen speziellen Puffer aus und überträgt das Programm - wie die Daten - mit dem 'Print#'-Befehl. Die C-1570/71 kann aber noch mehr. Es gibt spezielle Befehle, die nur dazu dienen, den Inhalt einer bestimmten Speicherstelle des Floppy-RAMs an den Rechner zu senden. Dieser Befehl wird 'memory-read' genannt. Seine Syntax sieht wie folgt aus:

"m-r"+chr$(l)+chr$(h)+chr$(z)

l = Low-Byte der Speicheradresse
h = High-Byte der Speicheradresse
z = Zahl der Bytes, die zu lesen sind

Die Parameter l und h ergeben die Adresse der gewünschten Speicherstelle. Mit z kann die Anzahl der Bytes, die man ab dieser Speicherstelle lesen möchte, bestimmt werden. Die Angabe z darf allerdings auch entfallen. Dann nimmt die Floppy an, daß nur ein Byte erwünscht ist. Der 'm-r'-Befehl wird über den Befehlskanal an die Floppy übermittelt. Über den gleichen Kanal werden die Daten dann auch eingelesen. Wenn Sie beipielsweise die Speicherstelle 151 (hex. $97) lesen möchten, könnte die Befehlsfolge beispielsweise lauten:

```
a = 151                                 Adresse festlegen
OPEN 1,8,15                             Befehlskanal öffnen
PRINT#1,"m-r"CHR$(a and255)CHR$(a/256)  Adresse an Floppy
GET#1,a$                                Byte von Floppy
PRINT ASC(a$+CHR$(0))                   Bytewert ausgeben
```

In diesem Programm würde die Zahl der Sektoren pro Spur des letzten IBM-34-Formats abgefragt.

Ebenso können an die Floppy Byte-Werte übermittelt werden, die dann in eine bestimmte Speicherstelle geschrieben werden. Der dazu notwendige Befehl lautet:

"m-w"+CHR$(l)+CHR$(h)+CHR$(z)+CHR$(b1)+CHR$(b2)...

Wie Sie sehen wird mit l und h wieder die Adresse der Speicherstelle angegeben. Danach folgt die Anzahl der Bytes, die ab dieser Speicherstelle in das Floppy-RAM geschrieben werden. Diesmal können Sie die Angabe z nicht weglassen. Zum Schluß folgen noch die eigentlichen Daten-Bytes. Dabei können bei einem 'm-w'-Befehl maximal 34 Bytes gleichzeitig übermittelt werden. Dies hängt wieder damit zusammen, daß der Eingabepuffer der C-1570/71 nur 41 Zeichen groß ist. Wenn Sie größere Speicherabschnitte, zum Beispiel ein Maschinenspracheprogramm, in das RAM schreiben wollen, müssen Sie dieses Vorhaben eben in mehreren Teilschritten erledigen.

3.1.2 MEMORY-EXECUTE UND BLOCK-EXECUTE

Das Einlesen eines Programms in den Puffer bringt natürlich nichts. Man muß dieses Programm auch irgendwie starten können. Das geschieht mit dem 'm-e'-Befehl. Der Befehl hat folgende Parameter:

"m-e"+chr$(l)+chr$(h)

Wieder muß eine Speicherstelle, zerlegt in Low- und High-Byte angegeben werden. Das Betriebssystem der C-1570/71 springt nun an diese Speicherstelle. Dabei muß an dieser Adresse ein sinnvolles Programm beginnen - ansonsten stürzt das Floppy-System ab. Wenn man in seinem Assembler-Programm die Anweisung 'RTS' erteilt, nimmt das Betriebssystem seine Arbeit auf.

Die Spezialisten unter Ihnenwissen jetzt natürlich, daß man auch ganz bestimmte Unterprogramme des Floppy-Betriebssystems aufrufen kann. Das folgende kleine Programm würde zum Beipiel eine bestimmte Spur total zerstören:

```
10 s = 18
20 OPEN 1,8,15
30 PRINT#1,"m-w"CHR$(0)CHR$(3)CHR$(6)
   CHR$(20)CHR$(163)CHR$(253)CHR$(76)CHR$(160)CHR$(234)
40 PRINT#1,"m-w"CHR$(6)CHR$(0)CHR$(1)CHR$(s)
50 PRINT#1,"m-w"CHR$(0)CHR$(0)CHR$(1)CHR$(224)
60 CLOSE1
```

Die Spur muß dabei in s angegeben werden. Verwenden Sie zu diesem Experiment unbedingt eine neu formatierte oder eine nicht mehr benutzte Diskette, denn die Daten werden nicht nur völlig zerstört, sondern das Betriebssystem wird bei dieser Diskette immer ziemlich durcheinandergeraten. Das Programm erzeugt nämlich einen sogenannten 'Killertrack' (siehe 6.3.5).

Sicherlich vermissen Sie in diesem Beispiel den 'm-e'-Befehl. Das Programm wird durch eine raffinierte Methode über den 'm-w'-Befehl gestartet. Das soll uns nicht weiter stören. Es kam auch

vielmehr darauf an, zu zeigen, welche Möglichkeiten Sie mit dem Speicherzugriff haben - selbst von BASIC aus.

Wenn man größere Programme in den Puffer bringen möchte, um sie dort auszuführen, kann das ganz schön zeitraubend werden. Am sinnvollsten wäre es, wenn man das Programm von Diskette in den Puffer einlesen würde und dann starten könnte. Genau gesehen müssen Sie jetzt den 'U1'- und den 'm-e'-Befehl kombinieren. Dabei wird der Inhalt eines Sektors (das Programm) in den Puffer gelesen und dann ausgeführt. Die Entwickler der Floppy haben sich überlegt, daß dies auch in einem Befehl möglich sein müßte. Dieser lautet:

"b-e k l t s"
"b-e";k;l;t;s

Dabei handelt es sich bei k und l um die Kanal- und Laufwerksnummer, wie Sie es schon bei den Direktzugriffsbefehlen kennengelernt haben. t und s sind wieder Spur- und Sektornummer. Der angewählte Sektor wird bei 'b-e' in den dem Kanal zugewiesenen Puffer geladen. Danach wird an den Anfang des Puffers gesprungen, um dessen Inhalt als Maschinenspracheprogramm auszuführen.

Der Befehl bringt gegenüber einer Kombination des 'U1'- und "m-e"-Befehls kaum einen Vorteil. Ausserdem wird er sowieso sehr selten verwendet. Wenn man nämlich Programme von Diskette in das Floppy-RAM lesen und dort ausführen möchte, gibt es dafür noch einen weiteren, wesentlich besseren Befehl, den wir in 3.1.5 besprechen werden.

3.1.3 Die User-Befehle

User-Befehle sind solche, die das Betriebssystem veranlassen, Programme an einer bestimmten Stelle im Speicher auszuführen. Dazu wird zuerst ein 'U' angegeben, danach folgt eine Ziffer oder ein Buchstabe. Dieses zweite Zeichen wählt eine aus mehreren vordefinierten Adressen aus, die angesprungen werden können.

Folgende User-Befehle sind vorhanden:

User-Befehl	Adresse	Funktion
U1 oder UA	$CD5F	Block-Read-Befehl
U2 oder UB	$CD97	Block-Write-Befehl
U3 oder UC	$0500	Sprung in Puffer 2
U4 oder UD	$0503	Sprung in Puffer 2
U5 oder UE	$0506	Sprung in Puffer 2
U6 oder UF	$0509	Sprung in Puffer 2
U7 oder UG	$050C	Sprung in Puffer 2
U8 oder UH	$050F	Sprung in Puffer 2
U9 oder UI	$FF01	1540/41-Bus umschalten
U: oder UJ	$EAA0	Reset
U; oder UK	$FE67	Interruptroutine

Einige der User-Befehle springen in Puffer 2 (U3-U8). Dabei haben die Adressen einen Abstand von genau 3 Bytes. So kann man in diesem Puffer sehr einfach eine Vektortabelle einrichten. Dies ist eine Liste mit lauter Sprungbefehlen, die dann zu den einzelnen Funktionen, die mit den User-Befehlen aufgerufen werden, verzweigt.

Die restlichen User-Befehle springen an verschiedene Stellen im Betriebssystem. Damit wurden noch einige zusätzliche Floppy-Funktionen eingebaut. Die Befehle U1 und U2 kennen Sie ja bereits aus 2.1.

Der U9- oder UI-Befehl dient dazu, zwischen dem 1540- und 1541-Bus umzuschalten. Die 1540 war die Floppy zum VC-20. Da der VC-20 eine etwas höhere Taktfrequenz als der C-64 hat, sollte man mit 'UI-' den Bus etwas schneller schalten. Die Befehlsfolge

'UI+' schaltet den Bus wieder auf das 1541-Zeitverhalten. Wird das '+' oder '-' weggelassen oder ein anderes Zeichen angegeben, dann führt die Floppy einen Teil-Reset durch. Dabei werden Zeropage und Systemzeiger neu eingerichtet. Ein RAM/ROM-Test sowie das Anlaufen des Laufwerkmotors entfallen aber.

Der 'UJ'-Befehl ist der Total-Reset. Dabei verhält sich die C-1570/71, als ob man sie einmal aus- und wieder eingeschaltet hätte.

Die C-1570/71 enthält (im Gegensatz zur 1541) noch den 'UK'-Befehl. Mit diesem Befehl wird ein 'BRK'-Befehl angesprungen (siehe ROM-Listing $AA2D). Folglich wird mit 'UK' die Interruptroutine gestartet. Im Normalbetrieb hat das keine besondere Wirkung. Hat man aber in diese Routine ein eigenes Programm eingefügt (mehr darüber im DOS-Kapitel), dann kann dieses damit gestartet werden.

Die User-Befehle haben gegenüber dem 'm-e'-Befehl einen gewaltigen Vorteil. Man kann sie in fast allen Situationen verwenden, wenn ein Programm nur eine Funktion zur Eingabe von Floppy-Befehlen hat - sei es die Textverarbeitung, das Datenbankprogramm usw.

Der 'm-e'-Befehl hingegen kann nur in BASIC angewendet werden, denn man benötigt die CHR$-Funktion, um Low- und High-Byte der Startadresse zu übermitteln.

3.1.4 DIE USER0-BEFEHLE

Es ist bei COMMODORE schon fast Tradition, daß in den Geräten viele interessante Funktionen enthalten sind, die im Handbuch schlichtweg verschwiegen werden. Und so bietet auch die C-1570/71 noch eine ganze Reihe von Befehlen, die überwiegend für die Handhabung von Disketten im CP/M-Format 'IBM System 34' zuständig sind.

Allen User0-Befehlen folgt eine Befehlsnummer. Diese Nummer setzt sich aus verschiedenen Bit-Daten zusammen. Sie wird des-

halb üblicherweise mit dem CHR$-Befehl in die Befehlskette eingebunden. Danach folgen die Parameter der einzelnen Befehle. Alle Befehlsnummern setzen sich aus folgenden Daten zusammen:

```
Bit0    : Laufwerksnummer (0/1)
Bit1-3  : Nummer der User0-Funktion
Bit4    : Angesprochene Diskettenseite
          0= Seite 1   1= Seite 2
Bit5-7  : Verschiedene Steuerflags
```

Die Laufwerksnummer lautet natürlich bei der C-1570/71 immer 0. Hier sind die User0-Befehle schon auf ein eventuell einmal erscheinendes Doppellaufwerk eingerichtet. Bei der C-1570 muß Bit 4 selbstverständlich auch immer auf 0 bleiben, da die C-1570 nur eine Diskettenseite verwenden kann.

Alle User0-Befehle funktionieren nur, wenn die Floppy im C-1570/71-Modus betrieben. Im 1541-Modus werden Sie ignoriert. Die einzige Ausnahme ist die Befehlsnummer 31. Hier werden unter anderem die Funktionen zur Verfügung gestellt, mit denen man Diskettenseiten anwählen kann u.ä.

Schauen wir uns also zuerst diese neuen Zusatzbefehle an. Bei allen Befehlen muß es heißen:

"U0"+CHR$(31)+"aa"
oder "U0>aa"

Anstelle der Zeichen 'aa' muß dann die entsprechende Zusatzfunktion eingesetzt werden. Es gibt dabei folgende neue Befehle:

```
aa   Funktion
--------------------------------------------------------
M1   Schaltet die Floppy in den 1571-Modus. Dabei wird das System
     mit 2 MHz Taktfrequenz betrieben. Damit
     können die C-1571-Eigenschaften auch im C-64-Modus
     verwendet werden.
M0   Schaltet auf 1541-Modus mit 1 MHz Taktfrequenz.
--------------------------------------------------------
H0   Aktiviert den Kopf auf Seite 1
H1   Aktiviert den Kopf auf Seite 2
     Der H-Befehl (Head) funktioniert nur im 1541-Modus.
--------------------------------------------------------
 Bit 7 im steuert bei 'M' und 'H' die Disketten-
 initialisierung :
 0= Diskette wird nach Befehl initialisiert
 1= Diskette wird nach Befehl nicht initialisiert
 Befehlnummer 31 bedeutet also 'mit Initialisierung',
 Nummer 159 'ohne Initialisierung'
--------------------------------------------------------
Rx   Setzt Zahl der Leseversuche in Zeropage-Adresse $6A.
     Der ASCII-Wert x wird direkt in $6A übernommen
     (genaue Funktion der Adresse siehe Zeropage-Listing).
--------------------------------------------------------
Sx   Setzt Sektorversatz bei COMMODORE-Disketten ($69)
--------------------------------------------------------
T    Testet die ROM-Prüfsumme.
--------------------------------------------------------
x    Der ASCII-Wert x wird als neue Geräteadresse der
     Floppy übernommen. x muß im Bereich 4-15 liegen.
--------------------------------------------------------
```

Eine weitere, besonders im C-128-Modus wichtige Funktion ist das Datei-Schnell-Laden. Wie Sie wissen ist die Ladegeschwindigkeit beim C-128 wesentlich höher als beim C-64. Dieses Schnell-Laden wird nicht mehr über Kanal 0 organisiert, sondern nur durch einen Befehl über den Befehlskanal aufgerufen. Danach werden die Daten mit dem schnellen Busmodus an den Rechner übertragen. Der Befehl hat folgende Syntax:

OPEN1,8,15,"u0"+CHR$(32)+"dateiname"

Auch hier steuert Bit 7 in der Befehlsnummer (32) wieder eine Sonderfunktion:

```
Bit 7 :   0= Dateityp wird auf PRG getestet.
          1= Dateityp wird nicht getestet.
             Alle sequentiellen Dateitypen werden geladen.
```

d auf PRG getestet.

Doch jetzt wollen wir uns mit den wichtigsten User0-Befehlen beschäftigen. Dies sind Befehle für den Betrieb der Floppy im CP/M-Modus. Dabei müssen Sie zuerst einige Zeropage-Adressen kennenlernen. Diese werden zum einen beim Programmieren in Maschinenspache benötigt, sind aber auch in BASIC-Programmen anwendbar.

```
Adresse    Funktion
------------------------------------------------------
$3C    60   logischer Sektorversatz bei Disketten im
            IBM-System-34 Format.
            Wird bei 'Sektor lesen/schreiben'
            verwendet.
------------------------------------------------------
$24    36   Header des letzten IBM-34-Sektors
------------------------------------------------------
$5E    94   Bit 0-3 = Nummer der aktuellen Fehler-
            rückmeldung. Dies ist genau der Wert, der
            normalerweise von der Jobschleife in
            Zeropage-Adresse $00-$05 gesetzt wird.
            Bit 7 :  1= Diskette ist im IBM-Format
                     0= Diskette ist im Commodore-
                        Format
------------------------------------------------------
$60    96   kleinste Sektornummer der Spur
------------------------------------------------------
$61    97   größte Sektornummer der Spur
------------------------------------------------------
$97   151   Zahl der Sektoren der Spur
------------------------------------------------------
```

Bei allen CP/M-Funktionen, die die Floppy unterstüzt werden die Daten im schnellen Busmodus übertragen. Dieser Übertragungsmodus kann aber nur in Maschinensprache programmiert werden. BASIC-Programme sind zu langsam die Daten zu übernehmen. Wird eine CP/M-Funktion mit dem entsprechenden User-0-Befehl aufgerufen, dann sendet die Floppy die Daten, ohne daß sie vom Rechner emfangen werden - sie gehen verloren.

Dies ist nicht sehr tragisch, da die CP/M-Befehle folgende Zusatzmöglichkeiten bieten:

```
Bit 5: 1= Sektor nicht in Puffer lesen/schreiben.
       0= Sektor von Diskette in Puffer lesen/schreiben.
Bit 6: 1= Lese-/Schreibfehler nicht beachten.
       0= Lese-/Schreibfehler melden.
Bit 7: 1= Puffer nicht an Rechner übertragen.
       0= Puffer an Rechner übertragen.
```

Die Bits 5-7 der Befehlsnummer steuern verschiedene Sonderfunktionen. So kann auch die unter BASIC störende Übertragung der Daten durch Bit 7 verhindert werden. Auf diese Weise wird ein IBM-34-Sektor nur in den internen Floppy-Puffer eingelesen. Die Übertragung zum Rechner kann durch die Direktzugriffsbefehle geschehen.

Dazu müssen Sie wissen, daß die Daten eines IBM-34-Sektors immer ab Adresse $0300 des Floppy-Speichers abgelegt werden (Puffer 0). Der Grund dafür liegt darin, daß IBM-34-Sektoren bis zu 1024 Bytes umfassen können und somit 4 Puffer belegen. Das bedeutet auch, daß Sie in diesem Fall 4 verschiedene Direktzugriffkanäle verwalten müssen.

Nun ist die Frage, wie erkannt wird, welche Sektorlänge die eingelegte Diskette hat. Weiter ist es bei CP/M-Disketten auch möglich Disketten mit unterschiedlichen Anzahlen von Sektoren zu beschreiben. Es muß also eine Möglichkeit geben die in das Laufwerk eingelegte Diskette zu analysieren, um die Daten des Diskettenformats zu ermitteln.

Dazu bietet die Floppy zwei Sonderfunktionen. Mit der einen kann der Header des nächsten Sektors gelesen werden. Dabei wird zuerst versucht einen IBM-System-34-Sektor zu lesen. Scheitert dies, probiert die Floppy, ob es sich um einen Sektor im COMMODORE-Format handelt. Das Ergebnis der Leseversuche ist in Zeropage-Adresse $5E gespeichert. Bit 7 gibt den eingelegten Diskettentyp an.

Bei IBM-34-Disketten können nun die Zeropage-Adressen $24-$29 gelesen werden, die das ID-Feld des IBM-Sektors enthalten. Auf diese Weise gibt beispielsweise s$27 Aufschluß über die Länge des Sektors.

Die weiteren Daten des Diskettenformats ermittelt eine zweite User-0-Funktion. Diese liest alle ID-Felder einer IBM-34-Diskette und errechnet folgende Angaben:

1. Befehlsstatusbyte ($5C).
2. Zahl der Sektoren der Spur ($97).
3. Nummer der Spur, die im Header eingetragen ist.
4. kleinste Sektornummer der Spur ($60).
5. Größte Sektornummer der Spur ($61)
6. Sektorversatz.

Diese Angaben werden in der oben genannten Reihenfolge im schnellen Busmodus an den Rechner übermittelt. Ein BASIC-Programm wäre also nicht fähig die Daten zu empfangen. In diesem Falll müßte man sie durch Direktzugriffsbefehle ('m-r') direkt aus dem Floppy-Speicher lesen.

Befehlsnummern der Analyse-Befehle:

```
Bit 76543210  Funktion
    000x0100  nächsten Sektor-Header lesen.
              x= Seitennummer.
    y00x1010  Spur analysieren.
              x= Seitennummer.
              y= 1= als 4. Zeichen angegebene Spur ansteuern.
                 0= Spur 0 ansteuern.
```

Die Spur-Analysefunktion ist von BASIC aus nicht zu starten, da der User-0-Befehl fehlerhaft arbeitet. Deshalb sollten Sie dazu folgende Sequenz über den Befehlskanal übermitteln:

"m-w"chr$(0)chr$(5)chr$(3)chr$(76)chr$(30)chr$(133)

Die Funktion wird mit "u3" aufgerufen. Weitere Beispiele zum Analysieren fremder Formate enthält das Kapitel 4.2.3.

Wenn Sie auf diese Weise die Floppy auf die IBM-34-Diskette eingestellt haben, können Sie nun mittels User-0-Direktzugriffsbefehlen einzelne Sektoren lesen oder schreiben.

```
Bit 76543210  Funktion
    abcx0000  Sektor lesen.
    abcx0010  Sektor schreiben.

    x= Seitennummer
    a= Puffer an Rechner übertragen.
    b= Fehler beachten.
    c= Puffer schreiben/lesen.
```

Wie Sie sehen, können bei den Sektorbefehlen die Zusatzfunktionen in den Bits 5-7 angegeben werden, die bereits besprochen wurden. Durch Bit 4 kann die gewüchnschte Diskettenseite bestimmt werden. Bei der C-1570 Floppy muß dieses Bit selbstverständlich immer den Wert 0 haben, da dieses Laufwerk nur eine Dskettenseite bearbeiten kann.

Die Parameter, der zu bearbeitenden Sektoren wird im Anschluß an die Befehlnummer an die Floppy über den Befehlskanal übermittelt. Dabei lautet ein Befehl nun:

"u0"+chr$(befehl)+chr$(Spur)+chr$(Sektor)+chr$(Anzahl)+chr$(neu)

Die Spur- und Sektornummer müssen als ASCII-Wert angegeben werden. Der folgende Parameter ermöglicht gleich mehrere Sektoren hintereinander zu lesen und an den Rechner zu übertragen, wobei die Nummer des nächsten Sektors durch Addition der aktuellen Sektornummer und des Sektorversatz ($3C) entsteht. Diese Funktion hat nur dann einen Sinn, wenn der schnelle Busmodus verwendet wird. Zum Schluß kann noch eine Spurnummer genannt werden, die nach dem Sektor-Befehl angesteuert werden soll. Auf diese Weise kann die Floppy schon die nächste Spur ansteuern, während der Rechner noch die letzten Daten verarbeitet.
Letztlich bietet die C-1570/71 noch eine Funktion, die bei vielen anderen Floppies nicht möglich ist, die Möglichkeit belliebige IBM-System-34 Formate zu formatieren. Die Syntax des User-0-Befehls lautet:

```
Bit 76543210  Funktion

    0iyx0110  IBM-34-Diskette formatieren.

    x= Seite bei der begonnen werden soll.
    y= Zahl der zu formatierenden Seiten.
       (0= 1 Seite; 1= 2 Seiten)
    i= 1= Spur-Index-Marke schreiben.
       0= Index-Marke nicht schreiben.
```

Parameter:

```
 4.Zeichen: Bit7: 1= IBM-System-34-Format.
                  0= COMMODORE-Format.
            Bit6: 1= angegebene Sektortabelle verwenden.
                  0= Sektortabelle aus erster Nummer und
                     Versatz erstellen.
            Bit0-5: kleinste Sektornummer der Spur.
 5.Zeichen: Sektorversatz -1
            Bei COMMODORE-Format: ID1
 6.Zeichen: Kennzeichen für Sektorlänge [1].
            Bei COMMODORE-Format: ID2
 7.Zeichen: letzte logische Spurnummer [39].
 8.Zeichen: größte Sektornummer der Spur [16].
 9.Zeichen: erste logische Spurnummer [0].
10.Zeichen: erste physikalische Spurnummer [0].
11.Zeichen: Leer-Byte, mit dem Sektoren gefüllt werden [229].
ab 12.Zeichen: Hier werden, wenn Bit 6 des 4. Zeichens gesetzt ist, die
Nummern der Sektoren genannt.
```

Wie Sie sehen ist die Formatierfunktion sehr komplex, wodurch allerdings auch sehr umfangreiche Formatiermöglichkeiten gegeben sind. Es gibt kein Format, daß Sie mit dieser User-0-Funktion nicht erstellen könnten. Es ist sogar möglich ein Diskettenformat zu formatieren, daß die Floppy nicht mehr analysieren oder lesen kann. Dies ist beispielsweisee der Fall, wenn jede Spur nur einen Sektor enthält, alle Sektoren der Spur die gleiche Nummer haben, o.ä.

Die vielfältigen Möglichkeiten dieses Befehls lassen sich auch in BASIC nützen. Dazu sollten Sie die BASIC-Programme des Kapitels 4.2.3 eingehender studieren, die die Anwendung der User-0-Befehle ausführlich demonstrieren.

3.1.5 AUTOSTART-FILES

Die Autostart-Files sind im Handbuch der C-1570/71 nur kurz erwähnt. Die Funktion und der Nutzen dieser Programmform wurde, wie bei COMMODORE üblich, verschwiegen.

Eine Autostart-Datei beim täglichen Umgang mit der Floppy kaum zu gebrauchen. dabei handelt es sich eine USR-Datei, deren Inhalt in einen beliebigen RAM-Bereich des Floppy-Speichers geladen wird. Das bedeutet, daß Sie diese Dateiform nur benötigen, wenn Sie eigene Programme im Floppy-Speicher ausführen möchten. Darüberhinaus sind die Autostart-Files nicht so einfach aufgebaut, wie Programmdateien. Lassen Sie sich jedoch nicht abschrecken, diese Floppy-Funktion einzusetzen.

AUFBAU EINES AUTOSTART-FILE

Byte	Funktion
0/1	Startadresse im RAM (Low-Byte/High-Byte).
2	Zahl der Datenbytes in diesem Sektor (max. 255).
3 - n	Datenbytes für Autostart-Programm.
n+1	Prüfsumme, die aus Byte 1 bis Byte n errechnet wird.

Autostart-Programme sind auf der Diskette wie sequentielle Dateien organisiert. Der Dateityp muß allerdings 'USR' lauten. Die User-Dateien (USR) werden genauso wie die seuqntiellen Dateien gehandhabt, mit dem Unterschied, daß als Dateityp 'u' angegeben wird. Sie können User-Dateien nur mit dem BASIC 3.0-Befehl eröffnen, da BASIC 7.0 diese Dateiform nicht unterstützt.

Der Aufbau einer Autostart-Datei ist nicht ganz einfach. Sie besteht aus beliebig vielen Blöcken, deren jeweilige Struktur in der obigen Tabelle dargestellt ist. Jeder dieser Blöcke, die in der User-Datei direkt aufeinander folgen, wird von der Floppy seperat bearbeitet. Selbstverständlich darf die User-Datei auch nur aus einem Autostart-Block bestehen.

In einem Autostart-Block wird zuerst die Startadresse genannt, bei der die Programmdaten des Blocks im Floppy-RAM abgespeichert werden sollen. Darauf ist die Anzahl der Bytes angegeben, die bei dieser Startadresse abgelegt werden. Ab Byte 3 müßen die Datenbytes des Programms angegeben werden. Danach folgt noch eine Prüfsumme, die aus der Startadresse, der Anzahl der Datenbytes und den Datenbytes selbst errechnet wird, indem man die Werte dieser Bytes addiert. Entsteht beim Addieren ein Übertrag, so wird dieser zur Prüfsumme gezählt.

Um größere Programme im Floppy-Speicher zu betreiben, müssen diese in 255 Bytes umfassende Teile zerlegt werden. Danach wird für jeden dieser Teile ein eigener Autostart-Block erstellt. Da dies sehr aufwendig ist, folgt unten ein Programm, daß diese Arbeit übernimmt. Es erstellt aus jeder Programmdatei eine Autostart-Datei. Die ersten beiden Bytes der Programm-Datei, die Startadresse, werden ebenso als Startadresse des Autostart-Files benützt.

```
 10 dim a$(255)
 20 input "programm-datei ";b$
 30 input "user-datei ";c$
 40 open 1,8,0,b$
 50 open 2,8,2,c$+",u,w"
 60 gosub 280
 70 on sgn(st) goto 260: a= asc(d$)
 80 gosub 280
 90 on sgn(st) goto 260: a= a+asc(d$)*256
100 print#2, chr$(a and255)chr$(a/256);
110 p=0
120 for n= 1to255
130 gosub 280
140 p= (257*(p+asc(d$))/256)and255
150 ifst and64then 190
160 if sgn(st) then 260
170 a$(n)= d$
180 next
190 print#2, chr$(n);
200 for m= 1ton
210 print#2, a$(m);
220 next
230 print#2, chr$(p);
240 a= a+n
250 on n/256+1 goto 270,100
260 print "diskfehler!"
270 close2: close1: end
280 get#1,d$: d$= left$(d$+chr$(0),1)
290 return
```

Die Autostart-Datei wird von der Floppy geladen und automatisch gestartet, wenn man

OPEN 1,8,15,"&dateiname"

eingibt. Dabei wird das Programm im Floppy-Speicher bei der Adresse des ersten Autostart-Blocks gestartet.

KAPITEL 4
DIE C-1570/71 und CP/M

4.1 Wie steuert CP/M die Floppy

4.1.1 BDOS und BIOS

Wenn Sie sich eingehender mit dem Betriebssystem CP/M beschäftigen, werden Sie sehr schnell auf die Begriffe BDOS und BIOS stoßen. Das BDOS, eine Abkürzung für 'Basic Disk Operating System', ist der Programmteil der Betriebssystems, der die Arbeit mit dem Floppylaufwerk steuert. Es ist dabei für die Verwaltung der Datein, für die Organisation des Directory usw. zuständig. Der zweite Teil, das BIOS(Basic Input Output System), ist für die Ansteuerung der Floppy, das Schreiben und Lesen der Daten auf die Diskette usw. verantwortlich.

Selbstverständlich kann hier keine vollständige Beschreibung dieser CP/M-Teile erfolgen - nicht einmal eine grundlegende Einführung. Diese Thematik ist derartig umfangreich, daß damit ein Buch, wie dieses Floppybuch, spielend gefüllt werden kann. Aus diesem Grund werden nur einige interessante Aspekte der Floppy-Programmierung unter CP/M betrachtet.

Das BDOS ist bei allen CP/M-Systemen identisch und verwaltet die Daten in Blöcken, die 128 Bytes umfassen. Es ist nur für die logische Verwaltung und die Handhabung der Daten zuständig. Weiter ist das BDOS der Betriebssystemteil, der dem Programmierer eine Vielzahl von Funktionen zur Floppyhandhabung zur Verfügung stellt.

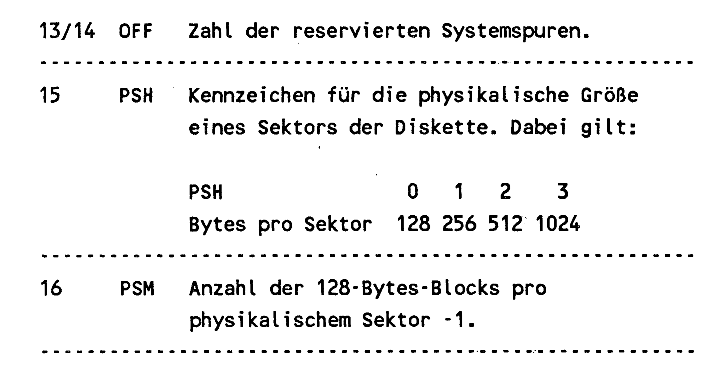
Das Schreiben und Lesen der Datenblöcke des BDOS übernimmt das BIOS. Dieser Teil steuert die einzelnen Laufwerke an. Deshalb wird das BIOS für jedes CP/M-System neu entwickelt, denn jedes Computersystem ist anders aufgebaut. So ist es Sache der Entwickler des Computersystems, wie leistungsfähig das BIOS ist. Es können beispielsweise mehrere verschiedene Diskettenformate verarbeitet werden usw.

4.1.2 DPB - Diskparameterblock

Zum Verwalten der Daten muß das BDOS die genauen Daten des Diskettenformats kennen. Schließlich ist es wichtig, welche Kapazität die Diskette hat oder wieviele Directoryeinträge möglich sind. Weiter muß das BIOS wissen, welche der Spuren der Diskette für Daten, für das Betriebssystem oder für das Directory verwendet weden. Ausserdem sind die Angaben über die Zahl der Sektoren pro Spur, Sektorversatz etc. wichtig.

Diese Daten des Diskettenformats werden in einer speziellen Tabelle verwaltet, die DPB (Diskparameterblock) genannt wird. Diese Tabelle enthält folgende Angaben:

Byte	Abk.	Funktion
1/2	SPT	Zahl der 128-Byte-Blöcke pro Spur
2	BSH	Block-Shiftfaktor. Diese Zahl gibt die Größe eines Verwaltungsblocks des BDOS an. Die einzelnen Blöcke des BDOS werden zu größeren Einheiten zusammengefaßt. Dabei gilt folgende Formel: Bytes pro Verwaltungsblock = 2 hoch (7+BSH) Es ergeben sich daraus folgende Werte: BSH 0 1 2 3 4 5 Blockgröße 128 256 512 1024 2048 4096 ...
3	BLM	Block Mask. Diese Zahl gibt die Anzahl der 128-Byte-BDOS-Blöcke pro Verwaltungsblock an. Der Wert ist dabei um eins vermindert, d.h. 7 bedeutet, daß 8 Blöcke enthalten sind.
4	EXM	Extend Mask.
5/6	DSM	Anzahl der 128-Byte Blöcke der Diskette (ohne Systemspuren) -1.
7/8	DRM	Anzahl der Directoryeinträge -1.
9 10	AL0 AL1	16-Bit-Belegungsverzeichnis, in dem festgelegt wird, welche Verwaltungsblöcke vom Directory belegt werden. Der erste Block der Directoryspur wird durch Bit 15 repräsentiert, der zweite Block durch Bit 14 usw.
11/12	CKS	Zahl der Directory-Einträge, die zum Erkennen eines Diskettenwechsels geprüft werden sollen.



13/14	OFF	Zahl der reservierten Systemspuren.
15	PSH	Kennzeichen für die physikalische Größe eines Sektors der Diskette. Dabei gilt:
16	PSM	Anzahl der 128-Bytes-Blocks pro physikalischem Sektor -1.

PSH	0	1	2	3
Bytes pro Sektor	128	256	512	1024

Das BIOS des C-128 CP/M+-Betriebssystems kann insgesamt 12 verschiedene Diskettenformate verarbeiten. Neben 3 COMMODORE-Formaten (C-64, C-128 einseitig, C-128 doppelseitig) werden auch 9 unterschiedliche IBM-34-Formate erkannt. Die DPB-Tabellen sind in der Datei 'CPM+.SYS' ab Adresse $1980 abgelegt, wenn man diese Datei mi einem Debugger, wie beispielsweise 'SID' oder 'DDT', einlädt. Beim Booten des Systems werden diese Tabellen in der ersten Bank des C-128 mit dem BIOS abgelegt. Aus diesem Grund können sie per Programm nur schwer erreicht werden, da der Programmspeicher durch die zweite 64K-Bank gebildet wird. Würde man auf die erste Bank umschalten ($3E oder $3F in $FF00), dann hängt sich der Computer auf, da sich das Programm damit ausblendet und auf diese Weise sozusagen Selbstmord begeht.

Jediglich die oberen 8K-Byte des Speichers werden nicht umgeschalten und enthalten immer den oberen Bereich der ersten 64K-Bank. Sie können diesen Bereich allerdings nur eingeschränkt nutzen, da er fast vollständig vom CP/M-Betriebssystem belegt wird.

Die Daten einer eingelegten Diskette, daß bedeutet die dazugehörige DPB-Tabelle kann mit der BDOS-Funktion $1F ermittelt werden. Dazu wird in A (Akku) die Nummer des Laufwerks angegeben und Sie erhalten nach dem Aufruf in HL

die Adresse des DPB. Die DPB-Tabellen der aktuellen Laufwerke liegen immer im oberen 8K-Block der Bank und können deshalb auch von einem Programm aus aufgerufen oder manipuliert werden.

4.2.1 MFM DATENAUFZEICHNUNG UNTER CP/M

Dieser Abschnitt handelt davon, wie Daten auf die Diskette geschrieben werden. Dabei dreht es sich nicht um die verwaltungstechnische Organisation der Daten. Was interessiert, ist die Technik, mit der die Elektronik Daten auf der Diskette aufzeichnet.

Dieses Aufzeichnungverfahren wird MFM genannt. Dies ist eine Abkürzung für 'modified frequency modulation'. Der Ausdruck 'modified' (engl. = modifiziert) weist schon darauf hin, daß es auch ein ein normales Aufzeichnungsformat gibt, welches 'FM' lautet.

Zuerst werden Sie das FM Verfahren kennenlernen, obwohl es von der C-1570/71 gar nicht verwendet wird, denn Sie haben es dann leichter, das MFM Verfahren zu verstehen.

Sicherlich wissen Sie, daß der Schreib-/Lesekopf eigentlich eine kleine Spule ist. Diese hat die Eigenschaft, je nach Stromdurchfluß, wie ein kleiner Magnet zu wirken, dessen Polarität, also die Ausrichtung des magnetischen Nord- und Südpols, davon abhängt, wie der durchfließende Strom gepolt ist. Das bedeutet, daß wir einen kleinen Magneten haben, den wir elektronisch immer wieder umdrehen können, je nachdem, welche Spannungspolarität angelegt wird.

Die Diskette besteht aus einem speziellen Material, welches magnetisiert wird. Dabei nimmt die Schicht die gleiche magnetische Polung an, wie der Magnet. Aus diesem Grund schaltet man den kleinen Magneten des Schreibkopfes elekronisch um und schreibt somit Informationen auf die Diskette. Eigentlich ganz einfach! Man magnetisiert die Diskette bei allen '0'-Bits in der einen Richtung und bei allen 1-Bits in der anderen.

Möchte man die Daten wieder lesen, erfolgt das auch mit der Spule des Schreib-/Lesekopfs. Sie liefert, je nach Polung der Magnetschicht der Diskette eine Spannung. Das passiert jedoch nur dann, wenn sich die magnetische Polung auf der Diskette ändert. Das bedeutet, ist die ganze Spur einer Diskette gleichgepolt, dann geschieht überhaupt nichts.

Deshalb geht man folgendermaßen vor: bei jedem '1'-Bit wird die Polung der Spule umgedreht, bei einem '0'-Bit nicht. Das Lesen ergibt dann, wenn eine '1' auf der Diskettte steht, einen kurzen Impuls am Lesekopf, da sich die Polarität auf der Diskette geändert hat. Geschieht nichts, dann handelt es sich um ein '0'-Bit.

Ein besonderes Problem ist der Motor des Floppylaufwerks. Dazu muß man wissen, das die Aufzeichnung eines einzelnen Bits auf der Diskette gerade einige millionstel Millimeter groß ist. Wenn der Motor dabei nicht äußerst ruhig läuft und aus Versehen einen winzig kleinen Ruck macht, ist schon ein Bit übersprungen worden.

Wenn man heute sogar Telefongespräche zum Mond leitet und wieder zurück, dann wird es es doch die Technik für etwas genauere Motoren geben? Aber natürlich! Möchten Sie aber für Ihre Diskettenstation mehrere hunderttausend Mark auf den Ladentisch legen?

Um die Laufwerkschwankungen auszugleichen, schreibt man Taktbits auf die Diskette. Ein Taktbit hat immer den Wert '1' und erzeugt sojedesmal einen Impuls am Lesekopf. Tritt dieser Impuls auf, dann weiß die Floppy-Elektronik, daß sie jetzt das Datenbit erwarten muß. Wenn in einer gewissen Zeit noch ein Impuls gelesen wird ist das Datenbit eine '1'. Fehlt dieser Impuls und es erscheint plötzlich schon das nächste Taktbit, dann muß es sich bei dem letzten Datenbit um eine '0' handeln.

Wie werden jetzt aber Takt- und Datenbit unterschieden? Bit ist doch Bit? Das ist richtig. Deshalb muß der Elektronik einmal mitgeteilt werden, daß es sich beim nächsten Bit um ein Taktbit handelt. Dann ist eine komplizierte Schaltung fähig, die Takt- und

Datenbits zu trennen. Wie die die Elektronik das Taktbit automatisch erkennt wird in 4.2.2 behandelt.

Ein Byte sieht nach dem FM Verfahren folgendermaßen aus:

```
T D T D T D T D T D T D T D T D
1 0 1 0 1 1 1 0 1 0 1 0 1 1 1 0

Datenbyte:

 0   0   1   0   0   0   1   0

T= Taktbit.
D= Datenbit.
```

Wenn Sie an Kapitel 1.1.2 zurückdenken, werden Sie sich sicher noch errinnern, daß einfach Bytes auf die Diskette zu schreiben, noch nicht genügt hat, schließlich wollen Sie die gespeicherten Daten auch wiederfinden. Das Problem war dabei den Beginn eines Datenblocks, also eines Sektors, zu kennzeichnen.

Dies macht man jetzt mit einer speziellen Markierung, die Sync-Zeichen genannt wird. Wie soll diese Markierung auf der Diskette aussehen, denn sie muß sich von den üblichen Aufzeichnungen unterscheiden. Dazu hat man sich folgenden Trick ausgedacht: es werden einfach ein paar bestimmte Taktbits weggelassen. Aber ist das nicht gefährlich? Was geschieht bei Motorschwankungen?

Aus diesem Grund wählt man als Werte für das Datenbyte zum Beipiel $FE. Bei diesem Wert kommen sehr viele Datenbits mit dem Zustand '1' vor, das bedeutet es sind auf der Diskette sehr viele Impulse vorhanden. Auf diese Weise findet sich die Elektronik beim Lesen zurecht und kann vor allem auch erkennen, daß das Taktbit fehlt. Um sich diese Sachverhalte klarer zu machen, trennt man den Wert für das Datenbyte und den für das Taktbyte. Denn wir könnten das Taktbit auch als Datenbit interpretieren und umgekehrt. Bei normalen Daten hat das Taktbyte immer den Wert $FF. Für jedes Bit ist das Taktbit '1'. Wird ein anderes Taktbytes verwendet, dann kann dieses Datenbyte anhand des Taktbytes von allen anderen Datenbytes eindeutig unterschieden werden.

Spezielle Taktbytes beim FM Verfahren:

```
T D T D T D T D T D T D T D T D
-------------------------------------------
1   1   0   0   0   1   1   1     Takt:  $C7
  1   1   1   1   1   0   0   0   Daten: $F8
-------------------------------------------
1   1   0   1   0   1   1   1     Takt:  $D7
  1   1   1   1   1   1   0   0   Daten: $FC
-------------------------------------------
1   1   0   0   0   1   1   1     Takt:  $C7
  1   1   1   1   1   1   1   0   Daten: $FE
-------------------------------------------
```

Normalerweise werden bei FM die Daten mit einer Rate von 250000 Bits pro Sekunde aufgezeichnet. Natürlich möchte man möglichst viele Daten auf der Diskette unterbringen. Der erste Genanke wäre, die Aufzeichnungsrate zu erhöhen, beispielsweise auf 500000 Bits pro Sekunde. Damit hätte sich die Kapazität der Diskette verdoppelt. Doch dabei gibt es physikalische Grenzen. Die Magnetschicht ist nicht fähig diese hohe Datenraten aufzunehmen. Denn bei 500000 Datenbits pro Sekunde werden neben den Datenbits auch noch 500000 Taktbits aufgezeichnet, summa summarum also 1000000 Impulse pro Sekunde. Es ist nicht möglich, derartig viele Impulse zu schreiben, da diese ineinander übergehen würden, weil Sie nicht genau genug aufgezeichnet werden können, sodas der zwischen zwei '1'-Impulsen eine Lücke vorhanden wäre.

Aus diesem Grund muß versucht werden, die Zahl der Impulse auf der Diskette zu verringern, ohne daß die Datenrate verändert wird.

Die Taktbits sind im Grunde genommen der störende Faktor, denn sie diehnen nicht zur Datenspeicherung, benötigen allerdings die Hälfte der Diskettenkapazität. Die Taktbits sind besonders wichtig, wenn das Datenbit den Wert '0' hat. In diesem Fall kann mit Hilfe des Taktbits erkannt werden, daß ein Datenbit fehlt. Hat das Datenbit den Wert '1', so werden Takt- und Datenbit durch einen Impuls dargestellt, wodurch die zu hohe Impulsrate

zustande kommt. Daher liegt es nahe bei allen Datenbits mit dem Zustand '1' die Taktbits wegfallen zu lassen und bei '0'-Bits die Taktbits zu schreiben. Mit dieser Methode entstehen zwischen den einzelnen Impulsen genügend große Abstände, die bei zwei aufeinander folgenden Takt- und Datenbits mit jeweils dem Wert '1' nicht vorhanden wären, da die Elektronik eine gewisse Anstiegs- und Impulsabfallszeit hat. Die Datenrate hat sich aller dings nicht verändert und beträgt nach wie vor 500000 Bits pro Sekunde.

Man kann sagen, daß für jedes Datenbit auf der Diskette eine Bitzelle vorhanden ist. Ist der Wert des Datenbits '0' dann wird ein Impuls am Beginn der Zelle aufgezeichnet, der Wert '1' wird durch einen Impuls in der Mitte der Bitzelle dargestellt. Bei Sync- und Index-Markierungen enthält eine Bitzelle des Datenbytes gar keinen Impuls und kann so als spezielle Makierung identifiziert werden.

4.2.2 DAS IBM SYSTEM 34 FORMAT

Mit 'IBM System 34' wird ein Diskettenformat benannt, daß sehr verbreitet ist. So gut wie alle Disketten-Controllerbausteine zeichnen die Daten nach diesem System auf. Das 'IBM System 34'-Format, von nun an kurz 'IBM-34' genannt, ist allerdings nicht die Art und Weise, wie die Daten auf der Diskette verwaltet werden, sondern die Methode, nach der auf der Diskette Spuren und Sektoren aufgebaut oder die Sync-Markierungen erstellt werden usw.

Im IBM-34-Format können Sektoren mit 128,=256, 512 und 1024 Bytes pro Sektor benützt werden, wobei die meisten Diskettenformaten Sektoren mit 256 Bytes Umfang verwenden. Aus diesem Grund werden wir nur den Aufbau einer Spur mit 256-Byte-Sektoren besprechen. Bei anderen Sektorgrößen wird das gleiche Prinzip der Sektorenaufzeichnung verwendet.

IBM-34-Disketten arbeiten immer mit dem Indexloch, das schon in Kapitel 1.1.2 erwähnt wurde. Dabei steuert diese Loch, an welcher Stelle die Sektoraufzeichnung auf der Spur beginnen soll.

Im Diagramm sehen Sie den Zustand der Indexloch-Lichtschranke oben eingezeichnet. Beim Auftreten des Indeximpulses werden auf der Spur 80 Bytes mit dem Wert $4E aufgezeichnet. Dieser Wert wird beim Formatieren als Füllwert für Lücken benützt. Durch diese Lücke nach dem Indexloch hat der Controller Zeit, die Schreib-/Leseelektronik zu aktivieren. Danach folgt der 'Pre-Index', eine Marke die aus 12 Bytes mit dem Wert $00 besteht. Bei diesem Wert werden am Schreib-/Lesekopf nur noch durch die Taktbits Impulse ausgelöst. Dadurch kann der Controller seine Leseelektronik so einstellen, daß sie bei normalen Datenbytes Takt- und Datenbits automatisch trennt. Die $00-Bytes dienen dazu dem Controller mitzuteilen, welches die Taktbits sind. Die Markierung mit $00-Bytes wird auch 'Sync' genannt, da der Controller damit synchronisiert wird.

Nach der Indexlücke folgt nun die sogenannte 'Index-Mark'. Dadurch wird dem Controller mitgeteilt, daß die vorhergehende Lücke zum Indexloch gehörte, da auch zwischen den einzelnen Sektoren Lücken vorhanden sind. Die 'Index-Mark' besteht bei MFM aus drei Bytes mit dem Wert $F6, gefolgt von einem $FC-Byte. Beim Wert $F6 wird beim Formatieren das Taktbyte $C2 verwendet. das bedeutet, daß zwischen dem dritten und vierten Datenbit das Taktbit fehlt, das an dieser Stelle normalerweise nötig wäre. Dadurch identifiziert der Controller die 'Index-Mark', da bei Datenbytes mit dem Wert $F6 dieses Taktbit nicht fehlt.

Im weiteren Verlauf der Spur ist nun eine Lücke mit 50 $4E-Bytes vorhanden. Somit hat der Controller Zeit, sich auf die Verarbeitung der Sektoren einzustellen. Nach dieser Lücke folgen 12 Bytes mit dem Wert $00, die eine Sync-Markierung darstellen. Die nächsten 3 Bytes haben den Datenwert $F5 und sind mit dem Taktbyte $A1 aufgezeichnet. Sie stellen zusammen mit dem darauf angegebenen $FE Byte, die 'ID Adress Mark' dar. Diese Markierung weist darauf hin, daß nun der Header des Sektors folgt. Die weiteren sechs Bytes sind der Sektorheader.

Zuerst wird die Spurnummer des Sektors genannt. Darauf folgt ein Byte, daß die Diskettenseite angibt. Dabei wird der Wert '0' für die Vorderseite und der Wert '1' für die Rückseite der Diskette verwendet.

Das nächste Byte ist die Sektornummer des nach dem Header folgenden Datenteils. Die vierte Angabe ist das Sektorkennzeichen, das die Größe des Datensektors angibt. Dabei haben die einzelnen Bytewerte folgende Bedeutung:

```
00  128 Bytes pro Sektor.
01  256 Bytes pro Sektor.
02  512 Bytes pro Sektor.
03 1024 Bytes pro Sektor.
```

Der Sektorheader wird mit zwei Prüfsummenbytes, auch CRC-Bytes genannt, abgeschlossen.

Dem Sektorheader folgt wieder eine 22 Bytes große Lücke mit dem Wert $4E, der sich 12 Bytes des Wertes $00 anschliesen, die eine Sync-Markierung darstellen.

Nun folgt die 'Data Adress Mark' die den Beginn des Datenbereichs kennzeichnet. Sie besteht aus 3 Bytes mit dem Datenwert $F5 und dem Taktbyte $A1, sowie einem Byte mit dem Wert $FB. Im Anschluß an die 'Data Adress Mark' sind die 256 Datenbytes des Sektors aufgezeichnet.

Zum Schluß werden zwei Prüfsummenbytes abgelegt. Diese werden nach dem CRC-Verfahren errechnet. CRC ist eine Abkürzung für 'cyclic redundancy check'. Bei dieser Methode wird aus den einzelnen Bits eines Datenbytes ein Polynom gebildet. Dieses Polynom wird durch ein Generatorpolynom, das G(x)=X16+X12+X5+1 lautet, dividiert. Normalerweise geht diese Division nicht auf und es entsteht ein Rest. Die CRC-Bytes sind der Wert, den man zum Polynom des Datenbytes addieren muß, damit sich bei der Division durch das Generatotpolynom keinen Rest ergibt. Dies hört sich kompliziert an, ist jedoch mit einer einfachen digitalen Schaltung lösbar.

Zum Schluß folgt wieder eine Lücke mit $4E-Bytes. Die Größe dieser Lücke hängt von der Sektorgröße ab. Darüberhinaus werden bei Laufwerken mit Gleichlaufschwankungen, die bis zu 3% betragen dürfen, größere Lücken verwendet, als bei relativ gleichlaufstabilen Laufwerken. Nach dieser Lücke folgt die Sync-Markierung vor der 'ID Adress Mark' des nächsten Sektors.

Wie eine Spur genau aufgebaut wird ist auch dem ROM-Listing zu entnehmen. Die Routine $8A86 formatiert eine Spur im IBM-34-Format. Aus ihr ist ersichtlich, wie welche Markierungen erzeugt werden, welchen Umfang die Lücken haben und mit welchen Parametern der Formatiervorgang gesteuert wird.

4.2.3 Fremde Disketten-Formate lesen

Eine der herausragenden Fähigkeiten der Floppy ist es fremde Diskettenformate zu verarbeiten. Dies wird nur im Betriebssysem CP/M ausgenutzt. Dabei erkennt das CP/M+ Betriebssystem des C-128 bereits verschiedene Formate von Epson, IBM, Kaypro und Osborne.

Vielleicht ist Ihr Anliegen ein anderes Format zu implementieren. Dazu gibt es mehrere Möglichkeiten. Sie können die Daten des Formats im BIOS verwerken und Ihr CP/M+ wird das Format immer automatisch erkennen. Die andere Möglichkeit wäre, das Diskettenformat durch Direktzugriffsbefehle (siehe 3.1.4) zu bearbeiten.

Bei beiden Anwendungen müssen allerdings die Daten des Formats bekannt sein. Dazu gehören Sektorlänge, Anzahl der Sektoren usw. Diese Angaben können mit den Analysefunktionen, die in 3.1.4 besprochen wurden, ermittelt werden. Da die Ermittlung des Aufzeichnungsverfahrens sehr umständlich ist, folgt nun ein kleines BASIC-Programm, das diese Arbeit übernimmt.
Achten Sie beim Eingeben darauf, die Leerzeichen und Steuerzeichen richtig zu setzen. Dadurch werden beim Programmablauf für die Eingabeparameter Standartwerte angegeben und Sie müssen jeweils nur die Return-Taste drücken, um einen Parameter zu übernehmen.

Zuerst fragt das Analyseprogramm nach der Geräteadresse und der Laufwerksnummer. Darauf kann die Nummer der Spur angegeben werden, die untersucht werden soll. Nach diesen Eingaben erfolgt die Analyse der eingelegten Diskette.

Zuerst wird festgestellt, ob es sich um ein IBM-34- oder ein COMMODORE-Format handelt. Das IBM-34-Format wird vom Programm einwandfrei erkannt. Ist jedoch kein IBM-Format und auch kein COMMODORE-Format vorhanden, die Diskette ist beispielsweise unformatiert, dann meldet sich das Programm trotzdem mit 'COMMODORE'. Sie sollten die Angabe eines COMMODORE-Formats grundsätzlich mit Vorsicht geniesen. Prüfen Sie in diesem Fall immer, ob sich auch ein ganzer Sektor oder das Directory lesen läßt.

Im Grunde genommen dient das Programm auch nur dazu IBM-34-Disketten zu untersuchen. Dabei werden zuerst einige Angaben aus dem Sektorheader aufgelistet. Das ist die im Sektorheader eingetragene Spurnummer, die Angabe über die Diskettenseite und das Sektorkennzeichen. Letzteres gibt an, wie lang die Sektoren der Spur sind.

Darauf folgen einige Daten, die die Floppy beim Lesen aller Sektor-ID-Felder errechnet hat. Dies ist die Anzahl der Sektoren der Spur und die kleinste und größte Sektornummer. Zum Schluß werden noch alle Sektornummer in der Reihenfolge aufgelistet, in der sie auf der Spur stehen. Mit dieser Liste können Sie den physiklischen Sektorversatz erkennen oder Unregelmäßigkeiten in der Sektorverteilung ausmachen.

Ebenso reizvoll wie die Analyse eines fremden Formates ist natürlich, Disketten in diesem Format zu formatieren. Dazu dient folgendes Programm:

Das Programm ermöglicht Ihnen die unterschiedlichsten IBM-System-34-Formate zu erstellen. Dazu müssen Sie zuerst Geräteadresse und Laufwerksnummer angeben. Danach folgen die Eingaben, die das Format festlegen.

Als erstes wird nach der Anzahl der zu formatierenden Spuren gefragt. Darauf muß die Nummer der Spur, die in die Sektorheader eingetragen werden soll, angegeben werden. Die nächste Eingabe legt fest, ab welcher physikalischer Spur formatiert werden soll. Damit können auch nur bestimmte Spurbereiche neu angelegt werden, sei es zur Reperatur einer beschädigten Stelle oder zur Verwirrung des Controllers.

```
1 dimn(32)
2 print"▒IBM System 34 Format▒▌"
3 input"Unit    8▌███▌";u
4 on1+(u>4)-(u>15)goto2
5 input"Side    2▌██▌";s:sd=(s/2)and1:s=sand1
6 input"▒Number of Tracks    40▌███▌";nt
7 input"log. Track-Start    0▌██▌";tl
8 input"pys. Track-Start    0▌██▌";tp
9 input"Sector-Size         3▌██▌";si
10 input"Number of Sectors   5▌██▌";sn
11 print"Define Sequenz (y/n)"
12 geta$:onasc(a$)and3goto13,17:goto12
13 sq=1:fora=1tosn
14 printa"▌▌. Sector   "right$(str$(a),2);
15 input"▌███▌";n:n(a)=nand31
16 next:goto19
17 sq=0:input "First Sector   1▌██▌";fs:fs=fsand31
18 input "Sector Skew    0▌██▌";sk:sk=((sk>0)*-sk)
   and31
19 input "Fill Byte      229▌████▌";by
20 b$="u0"+chr$(6+s*16+sd*32)+chr$(128+sq*64+fs)
21 b$=b$+chr$(sk)+chr$(si)+chr$(nt+tl-1)+chr$(sn)
22 b$=b$+chr$(tl)+chr$(tp)+chr$(byand255)
23 fora=1tosn:b$=b$+chr$(n(a)):next
24 print"▒Formatting..."
25 open1,u,15,b$:close1
26 ifds=0then28
27 print"▒▒▒Format Error▒"
28 print"▒▒One More Disk (y/n)"
29 geta$:onasc(a$)and3goto24,2:goto29

ready.
```

```
1 print"▓Format Analyser"
2 input"▓▓Unit      8▓▓▓▓";u
3 on1+(u>4)-(u>15)goto2
4 input"Drive    0▓▓▓▓";d:d=dand1
5 input"▓Track    0▓▓▓▓";tt
6 open1,u,15
7 print"▓Side 1      :";:s=0:gosub13
8 print"▓Side 2      :";:s=1:gosub13
9 print#1,"uj"
10 close1
11 print"▓1 next disk / 2 end"
12 geta$:onval(a$)+1goto12,5:end
13 print#1,"u0"chr$(158)"m1"
14 print#1,"u0"chr$(138+s*16)
15 a=94:gosub34:ifb<128then29
16 print#1,"m-w"chr$(0)chr$(5)chr$(3)chr$(76)
   chr$(30)chr$(133)
17 print#1,"u3-"+chr$(tt)
18 print" IBM System 34 Format"
19 a=36:gosub34:print"Tracknumber:"b
20 a=37:gosub34:print"Side-Bit    :"b
21 a=39:gosub34:print"Sector Size:"b;
22 printtab(17)"("2↑(7+b)"Bytes/Sektor )"
23 a=151:gosub34:n=b:print"nb. of Sec.:"b
24 a=96:gosub34:print"min. Sector:"b
25 a=97:gosub34:print"max. Sector:"b
26 print"Sequenz     :";
27 fora=523to522+n:gosub34:printb;:next:print
28 goto33
29 print" COMMODORE Format"
30 a=24:gosub34:print"▓Tracknumber:"b
31 a=22:gosub34:print"ID1 (DEC.) :"b
32 a=23:gosub34:print"ID2 (DEC.) :"b
33 return
34 print#1,"m-r"chr$(aand255)chr$(a/256)chr$(1)
35 get#1,a$:b=asc(a$+chr$(0))
36 return

ready.
```

Jetzt wird das Kennzeichen für die Sektorgröße verlangt, das Werte zwischen 0 und 3 haben darf. Die darauf folgende Frage erlaubt es die Abfolge der Sektornummern "von Hand" einzugeben und festzulegen. Wenn Sie dies nicht möchten müssen Sie hier mit 'n' antworten.

Sollte keine Sektor-Sequenz eingegeben worden sein, dann möchte das Programm nun die Nummer des ersten Sektors und den Sektorversatz wissen. Beim Sektorversatz handelt es sich um die Anzahl der Sektoren, die zwischen zwei aufeinander folgenden Sektoren eingebaut werden soll. Das Programm prüft nicht, ob die hier eingegebenen Werte einen Sinn ergeben. Dies stellt erst die Floppy fest und reklamiert eventuell mit dem Blinken der Fehleranzeige.

Zum Schluß können Sie noch das Byte definieren, mit dem die Sektoren gefüllt sind. Normalerweise ist die der Wert $E5 (229), wobei Sie darauf achten müssen, daß Sie hier keine Werte über $F0 (250) angeben, da diese beim Formatieren Steuerfunktionen haben.

Nun wird die Diskette formatiert. Wenn Sie bei der nach dem Formatieren folgenden Frage mit 'y' antorten, können Sie auf diese Weise weitere Disketten im gleichen Format erstellen, ohne die dazu nötigen Parameter neu einzugeben.

Die beiden BASIC-Programme sind nicht besonders umfangreich und nützten auch bei weitem noch nicht alle Möglichkeiten der Floppy aus. Sie sollen Ihnen vielmehr zeigen, wie die Floppyprogrammierung mit IBM-34 Disketten durchgeführt wird. Sie können dabei aus den BASIC-Programmen sicherlich einige Anregungen für Ihre eigenen Anwendungen entnehmen.

4.2.4 Der Controller WD 1770 - Programmierung.

Die Steuerung der IBM-34-Aufzeichnung wird in der C-1570/71 von einem eigenständigen Controller-Baustein ausgeführt - der WD 1770 von WESTERN DIGITAL.

In diesem Kapitel geht es darum, wie dieser Baustein angesprochen und programmiert wird. Dies kann allerdings nur in Mashinensprache und das auch nur im Floppy-Speicher erfolgen. Weitere Informationen über den technischen Aufbau des Controllers enthält das Kapitel 5.2.4.

Zur Steuerung des Controllers sind folgende Register vorhanden:

Adresse	Lesefunktion	Schreibfunktion
$2000	Status	Befehl
$2001	Spur	Spur
$2002	Sektor	Sektor
$2003	Daten	Daten

Wie Sie sehen hat das Register $2000 beim Lesen und beim Beschreiben unterschiedliche Funktionen. Wird ein Wert in diese Speicherstelle geschrieben, dann wird dieser als Kommando interpretiert. Beim Len des Registers erhält man allerdings nicht das Kommando, sondern eine Statusangabe, die den Betriebszustand des Controllers anzeigt. Die weiteren Register dienen dazu die Parameter der Kommandos festzulegen und Daten an den Controller zu übergeben oder vom Controller zu übernehmen.

Der Controller kennt folgende Kommandos:

Typ	Kommando	Kommandowert: Bit 7 6 5 4 3 2 1 0
1	Restore	0 0 0 0 h v x y
1	Seek	0 0 0 1 h v x y
1	Step	0 0 1 u h v x y
1	Step-in	0 1 0 u h v x y
1	Step-out	0 1 1 u h v x y
2	Read Sector	1 0 0 m h e 0 0
2	Write Sector	1 0 1 m h e p a
3	Read Adress	1 1 0 0 h e 0 0
3	Read Track	1 1 1 0 h e 0 0
3	Write Track	1 1 1 1 h e p 0
4	Force Interrupt	1 1 0 1 i j k l

Bedeutung spezieller Bits:

```
h:   0= Motor einschalten
     1= Motor ausschalten
v:   0= Spur überprüfen
     1= Spur nicht überprüfen
x/y: Stepprate  0 0 =  6ms
                0 1 = 12ms
                1 0 = 20ms
                1 1 = 30ms
u:   Spurregister nach Spur in Sektorheader setzen
     0= nein  1= ja
m:   0= nur ein Sektor lesen
     1= mehrere Sektoren lesen
a:   0= Data Marke für 'Sektor gültig' setzen
     1= Data Marke für 'Sektor gelöscht' setzen
e:   0= keine Kopfberuhigungszeit
     1= 30ms Kopfberuhigungszeit.
p:   0= Precompensation einschalten
     1= Precompensation ausschalten
```

i-l: Interruptbedingung
  i: nicht beachten
  j: nicht beachten
  k: Interrupt beim Auftreten des Index-Lochs
  l: Sofortiger unbedingter Interrupt
     bei i-l = 0 Befehl ohne Interrupt beenden.

Statusregister:

Bit0: Busy-Flag. Zeigt an, daß Kommando ausgeführt wird.
Bit1: Data Request/Index.

      Bei allen anderen Kommandos signalisiert dieses Bit,
      daß Daten aus Register $2003 entnommen oder
      in das Register geschrieben werden können.
Bit2: Lost Data/Track00.
      Bei Kommandos vom Typ 1 gibt dieses Bit an,
      daß der Kopf auf Spur 0 steht.
      Bei allen übrigen Kommandos wird durch dieses Bit
      angezeigt, daß die Daten des Registers $2003 vom
      Programm nicht rechtzeitig gelesen oder
      geschrieben wurden.
Bit3: CRC Error. Die Prüfsummenbytes des Headers oder des
      Datenblocks haben einen Fehler dekodiert.
Bit4: Record not found. Die angegebene Spur oder der Sektor
      wurde nicht gefunden.
Bit5: Spin-up/Record type.
      Bei Kommandos des Typs 1 wird durch dieses Bit
      angegeben, daß 6 Diskettenumdrehungen stattgefunden
      haben. Bei Kommandos des Typs 2 und 3 hat dieses
      Bit den Wert der 'Data Mark'.
Bit6: Write Protect. Diese Bit zeigt beim Schreiben an,
      wenn der Schreibschutz angebracht ist.
Bit7: Motor on. Dieses Bit gibt den Zustand des Motors an.
      0= Motor aus 1= Motor an.

Wie Sie sehen sind die Befehle des Controllers in verschiedene Kommandotypen unterteilt. Die verschiedenen Kommandotypen haben eine unterschiedliche Handhabung des Statusregisters und geben an, welche Parameterregister in welcher Art und Weise benutzt werden.

Einige Befehle oder Befehlsbits haben die Aufgabe den Stepper- und den Laufwerksmotor zu steuern. Diese Arbeit wird bei der C-1570/71 nicht vom IBM-34 Controller sondern vom Floppybetriebssystem erfüllt. Deshalb sind die Befehle des Typs 1 bei der COMMODORE-Floppy bedeutungslos.

Die Befehle des Typs 2 dienen zum Schreiben und Lesen einzelner Sektoren. Bevor einer dieser Befehle an das Befehlsregister übergeben werden kann, muß in Register $2002 zuerst die Nummer des gewünschten Sektors angegeben werden. Ist der gesuchte Sektor nicht vorhanden, so versucht der Controller fünfmal, den Sektor zu finden. Ist dies erfolglos, dann wird im Statusregister Bit4 gesetzt. Das Sektorregister $2002 gibt die Nummer des nächsten verfügbaren Sektors an.

Die Kommandos der Gruppe 3 dienen zum Berarbeiten gesamter Spuren und zum Analysieren der Spur. Das erste Kommando, der 'Read Adress' Befehl, liest den nächsten auftretenden Sektorheader und gibt in über das Datenregister $2003 aus. Dabei werden auch die beiden CRC-Bytes übermittelt. Das Statusbit 3 gibt an, ob diese Bytes richtig sind oder ob ein Prüfsummenfehler aufgetreten ist.

Das 'Read-Track' Kommando dient zum Lesen einer gesamten Spur, einschließlich der Adressmarken, der Lückenbytes usw. Die Luckenbytes können allerdings den falschen Wert haben, wenn sie zur Synchronsierung des Controllers dienen. Mit dieser Funktion kann eine gesamte Spur gelesen und analysiert werden.

Das Gegenstück dazu ist die 'Write Track' Funktion, die eine gesamte Spur beschreibt. Dieser Befehl wird zum Formatieren der Spur verwendet. Aus diesem Grund werden auch nicht alle Byte-Werte als Datenbytes auf die Diskette geschrieben. Die Werte von $F5 bis $F7 haben besondere Steueraufgaben:

$F5	ID Adress Mark. Schreibt $F5 mit Taktbyte $A1 (fehlendes Taktbit zwischen Bit 4 und 5).
$F6	Index Mark. Schreibt $F6 mit Taktbyte $C2 (fehlendes Taktbit zwischen Bit 3 und 4).
$F7	Schreibt anstatt des Bytes zwei CRC-Bytes auf die Diskette. Die Prüfsumme wurde mit den Daten seit der letzten Adress Marke errechnet.

Die Spurfunktionen beginnen das Lesen oder Schreiben beim Auftreten des Index-Lochs. Die Spur wird solange bearbeitet, bis das Index-Loch wieder auftritt und somit die Diskette eine Umdrehung vollzogen hat.

Leider ist es nicht möglich mit diesen beiden Befehlen gesamte Spuren einer Diskette auf eine andere Spur oder eine andere Diskette zu kopieren. Der Grund dafür liegt darin, daß beim Lesen bei den Lücken- und Synchronisationsbytes Fehler auftreten. Darüber hinaus, und das ist das schwerwiegendere Problem, würden die Datenbytes $F5-$F7 nicht als Datenbytes geschrieben, sondern als Kennzeichen für Adress- und Identifizierungsmarken verwendet.

Der Interruptbefehl dient zum Unterbrechen der aktuellen Funktion. Durch die Bits i-1 kann die Bedingung festgelegt werden, bei der der Befehl unterbrochen werden soll. Nach diesem Befehl muß mindestens 32 Mikrosekunden gewartet werden, bevor der Controller das nächste Kommando erhalten darf. Ansonsten würde er das aktuelle Kommando nicht unterbrechen.

KAPITEL 5
PROGRAMMIERUNG FÜR PROFIS

5.1 Wie Die Bytes auf der Diskette ausssehen...

5.1.1 Der Aufbau eines Sektors

Schon im Kapitel 1.1.2 wurde der prinzipielle Aufbau einer Diskette besprochen. Dabei haben Sie auch den Begriff des Sektors kennengelernt. Genau dieser Teil der Diskette wird im folgenden eingehender behandelt.

Wie schon bekannt ist, wird bei COMMODORE-Disketten der Beginn eines jeden Sektors mit einer speziellen Markierung versehen. Dadurch kann die Elektronik die Lage der Sektoren auf der Spur erkennen. Diese Markierung heißt Synchronisationsmarke, kurz 'Sync'.

```
-----------------------------------------------
-      -        -       -      -       -      -
- Sync - Header - Lücke - Sync - Daten - Lücke -
-      -        -       -      -       -      -
-----------------------------------------------
```

Die Abbildung zeigt die grundsätzliche Struktur eines Sektors. Der Sektor beginnt mit einer Sync-Markierung. Danach folgt der Sektorheader, der nachstehende Angaben enthält:

```
----------------------------------------------------------
-     -            -        -      -     -     -     -     -
- $08 - Prüfsumme - Sektor - Spur - ID2 - ID1 - $0F - $0F -
-     -            -        -      -     -     -     -     -
----------------------------------------------------------
```

Das erste Byte des Headers dient zur Identifizierung des Headers und hat den Wert $08. Dadurch stellt die Floppy fest, ob nach einer Sync-Markierung ein Header oder ein Datenfeld folgt. Der Datenteil beginnt mit dem Kennzeichen $07.

Nun folgt die Prüfsumme des Headers. Um sie zu errechnen werden Spur- und Sektornummer sowie die beiden ID-Zeichen

addiert. Die nächsten beiden Bytes des Headers enthalten Spur- und Sektornummer des Sektors. Mit dieser Angabe findet die Floppy einen gesuchten Sektor.

Zuletzt enthält jeder Sektorheader noch die beiden ID-Zeichen, die Sie beim Formatieren der Diskette angegeben haben. Diese Zeichen werden bei jedem Zugriff übernommen und geprüft. Haben sich die ID-Zeichen geändert, dann nimmt die Floppy an, daß die Diskette gewechselt wurde.

Die beiden Byte-Werte $0F haben keine Steuer-Funktion. Sie ergeben auf der Diskette die Bitfolge '0101010101', wodurch die Leseelektronik synchronisiert wird.

Dem Sektorheader folgt eine Lücke mit 9 Bytes Länge, bevor der eigentliche Datenteil beginnt. Diese Lücke dient dazu, daß man beim Schreiben genug Zeit hat, den Kopf auf Schreibbetrieb umzuschalten und zu aktivieren.

Darauf folgen die Daten des Sektors. Um deren Beginn exakt zu erkennen, geht dem Datenteil eine Sync-Markierung vorraus. Das erste Byte nach der Sync-Markierung hat den Wert $07. Damit kann der Datenteil vom Sektorheader unterschieden werden. Nach dem Datenkennzeichen sind die 256 Datenbytes des Sektors abgelegt. Zum Schluß ist wieder eine Prüfsumme angegeben, die sich aus der Summe der Datenbytes ergibt.

```
-----------------------------------
-      -     -            -       -
- Sync - $07 - Datenbytes - Prüfsumme -
-      -     -            -       -
-----------------------------------
```

Nach jedem Sektor ist wiederrum eine Lücke vorhanden. Deren Länge hängt von der Zahl der Sektoren der Spur und der Spurnummer ab. Sie wird aufgrund der Spurkapazität errechnet.

5.1.2 Die Sync-Markierungen

Wie Sie wissen, arbeitet die C-1570/71 Floppy beim Erkennen der Sektoranfänge bei COMMODRE-Formaten nicht mit dem Indexloch, sondern verwendet spezielle auf der Diskette aufgezeichnete Markierungen, die Sync genannt werden.

Diese Marken bestehen aus 5 Bytes mit dem Wert $FF, also 40 Bits mit dem Wert '1'. Die Leseelektronik erkennt, wenn mehr als 10 Bits mit dem Zustand '1' folgen und löst das Sync-Signal aus. Dieses Signal wird zum einen vom Betriebssystem der Floppy genutzt, wenn es auf den nächsten Sektor wartet, zum anderen weiß die Leseelektronik durch diese Markierung, wann die Datenbits eines Bytes beginnen, indem sie wartet, bis die '1'-Signale der Sync-Markierung vorüber sind.

Diese Sync-Markierung bereitet allerdings bei der C-1571 einige Probleme. Sicherlich haben Sie schon bemerkt, daß beim Booten des zum Computer mitgelieferten CP/M+ Betriebssystem die Floppy eine Zeit lang blinkt. Ausserdem geht der Vorgang gute 30 Sekunden schneller vor sich, wenn Sie das Betriebssystem auf eine neue Diskette kopieren, deren Rückseite unformatiert ist.

Der Grund für dieses Verhalten, daß beim Initialisieren der Diskette immer die Laufwerksanzeige blinkt und der Vorgang sehr lange dauert, sind die Sync-Markierungen der Rückseite. Die C-1571 versucht nämlich nach jedem Einlegen einer neuen Diskette zuerst festzustellen, ob beide Diskettenseiten beschrieben sind. Dazu führt sie auf beiden Seite Leseversuche durch. Beim Lesen orientiert sich die Floppy selbstverständlich an den Sync-Markierungen. Wenn Sie eine Diskette einlegen, die auf beiden Seiten beschrieben wurden, indem Sie die Diskette einseitig formatieren und dann umdrehen, geschieht folgendes: die Floppy liest auf der Rückseite, bis eine Sync-Markierung auftritt. Geschieht dies nicht innerhalb einer gewissen Zeit, dann geht die Floppy davon aus, daß die Rückseite unformatiert ist. Nun sind bei dem geschilderten Diskettentyp allerdings Sync-Markierungen vorhanden. Daß dabei die Diskette auf der Rückseite verkehrt herum läuft, da sie normalerweise gewendet wird, spielt keine Rolle, denn eine Folge von '1'-Werten, die Sync-Markierung, hat

vorwärts oder rückwärts gelesen die gleiche Wirkung. Nur die darauf folgenden Daten ist nicht der Sektorheader oder der Datenblock, sondern sind die Bytes einer Lücke.

Deshalb meldet die Leselogik einen Lesefehler. Der Haken ist nun, daß die Floppy nun eine Fehlerbehandlung einleitet. dabei wird das Lesen nochmals versucht, wobei der Kopf auch ein klein wenig neben der Spur positioniert wird. Genau dieser Vorgang, benötigt aber sehr viel Zeit.

Es ist also empfehlenswert, doppelseitig bespielte Disketten, die im "Wendeverfahren" benützt werden, auf ein einseitiges Format umzukopieren. Diejenigen C-64 oder VC-1541 Besitzer unter Ihnen, die auf diese Weise die Disketten besser ausnützen möchten, müssen bei der C-1571 in Kauf nehmen, daß der Initialisierungsvorgang etwas länger dauert.

Eine Lösung des Problems wäre auch mit dem User-0-Befehl 'U0>ra' möglich. Dieser setzt die Anzahl der Leseversuche, die bei einem Fehler durchgeführt werden sollen auf den Wert 1. Somit unterbleibt die aufwendige Fehlerroutine.

5.1.3 GCR-Kodierung, was ist das?

Sicherlich werden Sie sich schon gefragt haben, wie die Datenbytes bei COMMODORE-Formaten aufgezeichnet werden, denn Datenbytes mit dem Wert $FF würden unter Umständen als Sync-Markierung interpretiert.

Deshalb ist die Frage, wie die Daten bei COMMODORE-Disketten aufgezeichnet werden. Das Aufzeichnungsformat ist etwas exotisch, da auf unterschiedlichen Spuren verschiedene Aufzeichnungsraten verwendet werden. Das COMMODORE-Format ist somit weder in die Sparte der Single-Density-Formate, bei dehnen 250000 Bits pro Sekunde geschrieben werden, noch in die Reihe der Double-Density-Verfahren, die mit 500000 Bits pro Sekunde arbeiten, einzuordnen. Es wird nämlich eine Aufzeichnungsrate verwendet, die zwischen 250000 und 307692 Bits pro Sekunde schwankt. Dadurch das die äußeren Spuren einen größeren Um-

fang haben als die Inneren, kann man auf ihnen auch mehr Daten unterbringen. Deshalb gibt es bei COMMODORE-Disketten vier Spurzonen:

Spurnummer	Aufzeichnungsrate	Sektoren pro Spur
1 - 17	38461 Bytes/sec	21
18 - 24	35714 Bytes/sec	19
25 - 30	33333 Bytes/sec	18
30 - 35	31250 Bytes/sec	17

Zum Aufzeichnen der Daten wird das GCR-Verfahren verwendet, was soviel wie Group Code Recording bedeutet. Bei dieser Methode werden 4 Datenbits in 5 GCR-Bits umgewandelt. Ein Datenbyte, das 8 Bits umfaßt, wird somit durch 10 GCR-Bits dargestellt. Dazu zerlegt man das Datenbyte in zwei Hälften, den niederwertigen Teil (Bits 0-3) und den höherwertigen Teil (Bits 4-7). Die Bits jedes dieser Teile werden dann nach folgender Tabelle umgewandelt:

Dezimal	Binärbyte	GCR-Code
0	0000	01010
1	0001	01011
2	0010	10010
3	0011	10011
4	0100	01110
5	0101	01111
6	0110	10110
7	0111	10111
8	1000	01001
9	1001	11001
10	1010	11010
11	1011	11011
12	1100	01101
13	1101	11101
14	1110	11110
15	1111	10101

Diese GCR-Werte sind so gewählt, daß nach maximal 4 Bits mit dem Wert '1' eine Null geschrieben wird. Dadurch können Daten-

bytes auf der Diskette nach der Umrechnung in GCR-Bytes niemals eine Folge von mehr als 6 Einsen ergeben und somit auch nicht als Sync-Markierung interpretiert werden. Außerdem kommen bei den GCR-Werten niemals mehr als 2 Bits mit dem Zustand '0' hintereinander vor. Dies ist sehr wichtig, da die Leseelektronik anhand der '1'-Bits Laufwerksschwankungen ausgleicht.

Das Umrechnen der Daten wird mit Disketten-Betriebssystem immer in Gruppen zu jeweils vier Datenbytes vorgenommen. In diesem Fall besteht das Ergebnis aus genau 5 Bytes mit den entsprechenden GCR-Werten. Das Umwandeln der Daten geschieht aber nicht automatisch, sondern muß von einem Programm durchgeführt werden. Das DOS enthält dazu Routinen, die die Umwandlung mitels eines Algoritmus oder mit Hilfe von Tabellen vornehmen. Die erste Methode hat den Nachteil, daß das Programm aufwendiger ist und nur langsam arbeitet, die Tabellenmethode hingegen benötigt viel Speicherplatz, ist dafür aber etwas einfacher und schneller.

Diese Umrechnung der Binärdaten in GCR-Werte und umgekehrt ist übrigens ein Grund, warum die Floppy ein eigenes Mikroprozessorsystem mit Pufferspeicher benötigt. Die Daten können nämlich nicht so schnell umgewandelt werden, wie sie auf die Diskette geschrieben werden müssen (siehe Aufzeichnungsrate). Die Daten werden zwischengespeichert, umgewandelt und danach auf die Diskette transferiert.

Zum Schluß noch einige Beipiele, wie 4 Binärbytes in GCR-Werte umgerechnet werden:

```
Datenbytes $01       $02       $03       $04
Binärwert  0000 0001 0000 0010 0000 0011 0000 0100
GCR-Wert   0101001011010101001001010100110101001110
GCR-Bytes  $52     $C5     $25     $4C     $4E

Datenbytes $A1       $FC       $65       $9D
Binärwert  1010 0001 1111 1100 0110 0101 1001 1101
GCR-Wert   1101001011101010110110110011111100111101
GCR-Bytes  $D2     $EA     $DB     $3F     $3D
```

5.2. Wer die Bytes auf die Diskette bringt . . .

5.2.1. 1570/71 Schaltschema

Die folgenden Kapitel behandeln das Steuersystem der Floppy. Dabei geht es aber vorwiegend darum, wie bestimmte elektronische Bausteine bei der C-1570/71 eingesetzt sind und welche Aufgaben sie erfüllen. Eine allgemeine Einführung in die Mikroprozessortechnik können diese Kapitel selbstverständlich nicht bieten. Auch können die in der C-1570/71 verwendeten Bausteine nur in Bezug auf Ihre Tätigkeit in der Floppy besprochen werden. Die grundsätzliche Programmierung und Beschaltung dieser Bauelemente müssen Sie entsprechenden Datenblättern entnehmen.

Das Schaltschema der Floppy kann natürlich keinen vollständigen Schaltplan ersetzen, es soll Ihnen nur den Aufbau einiger wichtiger Elemente der Floppy verdeutlichen. Der Kern des Mirkocomputers ist ein Prozessor vom Typ 6502B. Dieser kann mit einer Taktrate von bis zu 2MHz betrieben werden. Bei der C-1570/71 kann zwischen 1MHz und 2MHz Takt umgeschalten werden. Im 1541 Modus benützt die Floppy die langsamere Prozessorfrequenz, da die C-1541 auch mit dieser Frequenz arbeitet. Ist die Floppy im C-1571 Modus, dann wird der Prozessor mit 2MHz getaktet. Der Grund für die unterschiedlichen Taktfrequenzen sind die Busroutinen. Bei diesen Programmteilen kommt es auf den zeitlichen Ablauf der Bussignale an, daß die Floppy schnell genug reagiert, und daß die Daten in den richtigen Abständen ausgegeben werden.

Die höhere Taktfrequenz von 2MHz hat einige Vorteile. Die Daten, die von der Diskette gelesen werden, können schneller verarbeitet werden. Dies betrifft beispielsweise die GCR-Umwandlung, denn nun ist es möglich ein Byte sofort nachdem es gelesen wurde von GCR nach Binär umzurechnen. Darüberhinaus kann der Bus nun mit einer maximalen Übertragungsrate von 500000 Baud betrieben werden. Bei dieser Geschwindigkeit ist es durchaus möglich eine gesamte Spur sofort während des Lesens an den Rechner zu übertragen. Daß diese hervoragenden Fähigkeiten der Floppy-Elektronik nicht genutzt werden liegt allein am Betriebssystem der C-1570/71.

An den Prozessor sind drei Ein-/Ausgabebausteine, ein IBM-34-Format-Controller, 2KByte RAM und 32KByte ROM angeschlossen. Diese einzelnen Komponenten belegen folgende Adressbereiche:

Bereich	Belegung
\$0000 - \$07FF	2KByte RAM
\$1800 - \$180F	6522 (VIA1) Steuert Bus und 1571-Elektronik
\$1C00 - \$1C0F	6522 (VIA2) Steuert Aufzeichnungselektronik, Motor usw.
\$2000 - \$2003	WD 1770 Steuert IBM-34-Aufzeichnung
\$4000 - \$400F	6526 (CIA1) Steuert schnellen Bus-Modus
\$8000 - \$FFFF	32KByte ROM Betriebssystem

5.2.2 Die Interfacebausteine

Dieses Kapitel behandelt die Interfacebausteine der Typen 6522 und 6526. Zum besseren Verständnis dieser Schaltkreise werden die Datenblätter zum 6522, die von vielen Halbleiterhändlern erhältlich sind, empfohlen. Zum 6526 gibt es im freien Handel leider keine Unterlagen, da dieser Baustein eine Eigenentwicklung von COMMODORE ist. Ausführliche Informationen uber den 6526 finden Sie in den INTERN-Büchern zum C-64 und C-128.

Zuerst soll auf den 6522-Baustein eingegangen. Dieser wird auch VIA (Versatile Interface Adapter) genannt. Die 1570/71 hat zwei derartige Bausteine eingebaut. Das VIA hat 40 Anschlüße, die folgendermaßen belegt sind (nur die wichtigsten):

Anschluß	Name	Funktion
Pin 2-9	PA	8-Datenleitungen, die frei programmiert werden können.
Pin 10-17	PB	8-Datenleitungen, die frei programmiert werden können.
Pin 18	CB1	Steuerleitung.
Pin 19	CB2	Steuerleitung.
Pin 26-33	D7-0	8 Datenleitungen zum Prozessor.
Pin 39	CA1	Steuerleitung.
Pin 40	CA2	Steuerleitung.

Die einzelnen Steuer- und Datenleitungen des VIA werden vom Computer angesteuert. Dazu hat das VIA 16 Register, die im Speicherbereich des Prozessors liegen, über die der Prozessor den Ein-/Ausgabebaustein steuern kann, indem er bestimmte Werte in diese Register schreibt. Darüber hinaus sind in dem VIA auch noch zwei Zähler eingebaut. Diese Zählen die Taktimpulse des Prozessors. Haben die Zähler einen bestimmten Wert erreicht, können verschieden Aktionen ausgelöst werden. Im Grunde genommen kann man auf diese Art und Weise eine bestimmte Zeitspanne programmieren, nach der ein Signal erzeugt werden soll. Deshalb werden diese Zähler meistens Timer (engl. = Zeitgeber) genannt.

Registerbelegung des VIA 6522

Adresse	Funktion
n	Datenregister für PB.
n + $01	Datenregister für PA mit Handshaking.
n + $02	Datenrichtungsregister für PB.
n + $03	Datenrichtungsregister für PA.
n + $04	Low-Byte Timer 1.
n + $05	High-Byte Timer 1.
n + $06	Ausgangswert von Timer 1 (Low-Byte).
n + $07	Ausgangswert von Timer 1 (High-Byte).
n + $08	Low-Byte Timer 2.
n + $09	High-Byte Timer 2.
n + $0A	Serielle Ein-/Ausgabeleitung.
n + $0B	Hilfs-Steuer-Register.
n + $0C	Peripheres Steuer-Register.
n + $0D	Unterbrechungs-Flag-Register.
n + $0E	Unterbrechungs-Maske.
n + $0F	Datenregister für PA (ohne Handshaking).

n= für VIA1: $1800
   für VIA2: $1C00

Das VIA hat zweimal 8 Ein-/Ausgangsleitungen. Welche Leitung als Ausgang und welche als Eingang verwendet wird, wird im Datenrichtungsregister festgelegt, wobei jedes Bit des Registers einer Leitung entspricht. Hat ein Bit den Wert '0', dann wird die dazugehörige Leitung als Eingang verwendet, beim Wert '1' fungiert der Anschluß als Ausgang. Beim Eingangsbetrieb wird das anliegende Signal in das entsprechende Bit des Datenregisters übernommen. Ist die Datenleitung als Ausgang geschlaten, wird der Pegel des entsprechenden Bits im Datenregister ausgegeben.

Die zwei Ausgangsports (port= 8 Datenleitungen) werden PA und PB genannt. Der Port PA hat zwei verschiedene Datenregister. Arbeitet man mit dem Datenregister in $01, dann werden immer beim Schreiben eines neuen Wertes in dieses Register die Steuerleitungen beeinflußt. So kann beispielsweise über die Steuerleitung

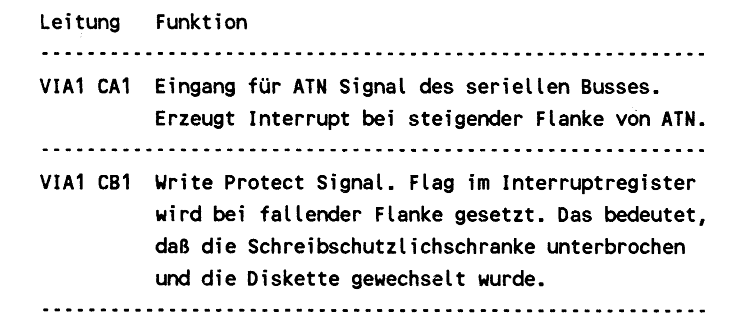
ein Impuls ausgegeben werden, womit die angeschlossene Logik erkennt, daß ein neues Signal am Port anliegt. Diese Funktion wird aber bei der C-1570/71 nicht benutzt. Es ist also unerheblich, welches der beiden Datenregister Sie benützen.

Neben den Ein-/Ausgabeleitungen gibt es noch die Steuerleitungen CA und CB. Diese haben zum einen, wie schon erwähnt wurde, Steuerfunktionen beim Beschreiben des Datenregisters. Auf der anderen Seite können CA und CB auch als normale Ein-/Ausgabeleitungen benutzt werden. Dies ist deren Aufgabe in der C-1570/71. In welchem Modus die Steuerleitungen betrieben werden oder welchen Pegel sie haben wird im Peripheren Steuerregister festgelegt:

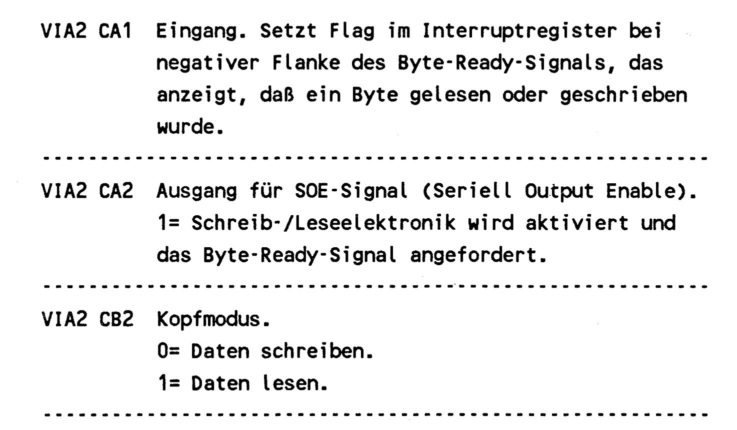
Peripheres Steuerregister:

```
Bit0  : 0   = CA1 Eingang Interrupt bei fallender Flanke.
        1   = CA1 Eingang Interrupt bei steigernder Flanke.
Bit1-3: 110= CA2 Ausgang mit Low-Pegel.
        111= CA2 Ausgang mit High-Pegel.
Bit4  : 0   = CB1 Eingang Interrupt bei fallender Flanke.
        1   = CB1 Eingang Interrupt bei steigernder Flanke.
Bit5-7: 110= CB2 Ausgang mit Low-Pegel.
        111= CB2 Ausgang mit High-Pegel.
```

Die Steuerleitungen der beiden VIAs der Floppy werden zu folgenden Zwecken verwendet:



Leitung	Funktion
VIA1 CA1	Eingang für ATN Signal des seriellen Busses. Erzeugt Interrupt bei steigender Flanke von ATN.
VIA1 CB1	Write Protect Signal. Flag im Interruptregister wird bei fallender Flanke gesetzt. Das bedeutet, daß die Schreibschutzlichschranke unterbrochen und die Diskette gewechselt wurde.



VIA2 CA1	Eingang. Setzt Flag im Interruptregister bei negativer Flanke des Byte-Ready-Signals, das anzeigt, daß ein Byte gelesen oder geschrieben wurde.
VIA2 CA2	Ausgang für SOE-Signal (Seriell Output Enable). 1= Schreib-/Leseelektronik wird aktiviert und das Byte-Ready-Signal angefordert.
VIA2 CB2	Kopfmodus. 0= Daten schreiben. 1= Daten lesen.

Die Steuerleitungen werden besonders in VIA2 benutzt. Dabei durch sie die Schreib-/Leseelektronik gestetzt oder es werden Rückmeldungen angenommen. Das wichtigste Signal dieser Art ist der Byte-Ready-Eingang. Dieses Signal zeigt an, wenn die Schreib-/Leselogik ein Byte verarbeitet und auf die Diskette geschrieben hat oder wenn ein Byte von Diskette gelesen wurde und nun zur weiteren Verarbeitung Verfügung steht. Das Byte-Ready-Signal wird auch noch an den Eingang PA7 des VIA1 und an den Eingang SO (Set Overflow) des Prozessors geleitet. Diese beiden letzten Eingänge werden vom Betriebssystem der Floppy benützt und abgefragt. Im 1571-Modus ist es PA7, im 1541-Modus wird SO verwendet. Ein High-Pegel an SO hat zur Folge, das das Overflow-Flag des Prozessors gesetzt wird. Auf diese Weise kann mit den 6502-Befehlen 'BVC' oder 'BVS' das Byte-Ready-Signal sehr komfortabel verarbeitet werden.

Die wichtigsten Aufgaben der beiden VIA-Bausteine werden jedoch nicht mit den Steuerleitungen, sondern mit den Ports PA und PB erledigt. Deshalb nun die Belegung dieser Ein-/Ausgabeleitungen:

Leitung	E/A	Funktion
VIA1 PB0	E	Data Eingang vom seriellen Bus.
VIA1 PB1	A	Data Ausgang auf seriellen Bus.
VIA1 PB2	E	Clock Eingang vom seriellen Bus.
VIA1 PB3	A	Clock Ausgang auf seriellen Bus.
VIA1 PB4	A	0= ATN wird automatisch beantwortet.
		1= ATN wird erzeugt keine autom. Antwort.
VIA1 PB5	E	DIP-Schalter 1 (links).
VIA1 PB6	E	DIP-Schalter 2 (rechts).
VIA1 PB7	E	ATN Signal vom seriellen Bus.
VIA1 PA0	E	Zustand der Spur0-Lichtschranke:
		0= Kopf ist auf Spur 0.
		1= Kopf ist nicht auf Spur 0.
VIA1 PA1	A	1570/71-Bus Datenrichtung.
		0= 1570/71 Bus ist auf Eingang.
		1= 1570/71 Bus ist auf Ausgang.
VIA1 PA2	A	aktiver Kopf (nur bei 1571).
VIA1 PA5	A	Floppy-Modus und Takt des Prozessors.
		0= 1541-Modus mit 1MHz Taktfrequenz.
		1= 1571-Modus mit 2MHz TAktfrequenz.
VIA1 PA7	E	Byte-Ready-Signal
VIA2 PB0	A	STP1.
		2. Bit der Steppersteuerung.
VIA2 PB1	A	STP0.
		1. Bit der Steppersteuerung.
VIA2 PB2	A	0= Laufwerksmotor aus.
		1= Laufwerksmotor ein.
VIA2 PB3	A	0= Laufwerksanzeige (LED) aus.
		1= Laufwerksanzeige (LED) ein.
VIA2 PB4	E	Zustand des Write Protect
		0= Diskette ist schreibgeschützt.
		1= Diskette ist beschreibbar.
VIA2 PB5	A	DS0.
VIA2 PB6	A	DS1.
		Die Signale DS0 und DS1 steuern die
		Aufzeichnungsrate auf der Diskette.
		(Funktion siehe ROM-Listing $9409)

```
VIA2 PB7  E    Sync-Signal
               1= Sync-Markierung ist aufgetreten
------------------------------------------------------
VIA2 PA   A    Daten, die an Schreib-Elektronik übergeben
               werden.
VIA2 PA   E    Daten, die von Lese-Elektronik gelesen
               wurden.
------------------------------------------------------
```

Ein weiterer Interface-Baustein, der in der Floppy verwendet wird, ist die CIA 6526. Diese Ein-/Ausgabeeinheit wird auch im C-64 und C-128 verwendet. Der CIA-Baustein ist dem 6522 sehr ähnlich. Er enthält allerdings noch zusätzlich eine Echteituhr mit Alarm.

In der C-1570/71 werden nur der serielle Ein-/Ausgang (SP) und die dazugehörende Taktleitung CNT verwendet. Diese beiden Leitungen sind für die Übertragung der Daten im schnellen Busmodus zuständig. Darüber erfahren Sie mehr in Kapitel 5.2.6.

Selbstverständlich hat auch die CIA 6526 zwei 8-Bit große Ausgangsports, die allerdings nicht verwendet werden. Sie haben somit die Möglichkeit, diese Leitungen für eigene Anwendungen und Basteleien zu verwenden.

5.2.3 Der Controller WD 1770 - Technik

Der WD 1770 wird von WESTERN DIGITAL hergestellt und ist softwarekompatibel zur WD 179X-Reihe, die auch von anderen Herstellern produziert wird. Dieser 28-polige Baustein enthält alles, was zur Steuerung eines Diskettenlaufwerks nötig ist. Dies ist beispielsweise eine Logik, die den Steppermotor und somit den Schreib-/Lesekopf bewegt. Darüberhinaus sind bereits alle Komponenten, die zum Lesen und Schreiben der Daten nötig sind, im WD 1770 intergriert.

Pin-Belegung des WD 1770:

Pin	Name	Funktion
1	CS	Chip Select. Ein Low-Signal an diesem Pin adressiert den Baustein.
2	RW	0= Register schreiben. 1= Register lesen.
3/4	A0/1	Adressleitungen, die bei CS=0 das gewünschte Register auswählen.
5-12	D0-7	Datenbus zum Prozessor
13	MR	Ein Low-Pegel führt den Reset durch.
14	GND	Masseanschluß.
15	Vcc	+5 V Spannung
16	STEP	Ausgang für Steppimpulse für Kopfmotor
17	DIRC	Direction. 0= Kopf bewegt sich nach außen. 1= Kopf bewegt sich nach innen.
18	CLK	Eingang für Betriebstakt von 8MHz.
19	RD	Read Data. Impulse von der Diskette. Diese Infomationen enthalten sowohl die Takt- als auch die Datenbits.
20	MO	Motor On. Ausgang um Motor anzuschalten.
21	WG	Write gate. Dieser Ausgang wird High, wenn auf die Diskette geschrieben wird.
22	WD	Dies sind die Daten zusammen mit den Taktbits, die auf die Diskette geschrieben werden.
23	TR00	Track0-Eingang: 0= Kopf ist auf Spur 0 1= Kopf ist nicht auf Spur 0.
24	IP	Index-Puls: 0= Index-Lichtschranke wurde unterbrochen. 1= Lichtschranke ist nicht unterbrochen.
25	WPRT	Write Protect: 0= Diskette ist schreibgeschützt. 1= Diskette ist beschreibbar.
26	DDEN	Double Density. 0= Double Densitiy 1= Single Sensity
27	DRQ	Data Request. 1= Datenregister ist bereit.
28	INTRQ	Interrupt Request. 1= Kommando beenden.

Die Steppersteuerung des WD 1770 wird in der C-1570/71 Floppy nicht verwendet. Dies übernimmt die Betriebssoftware über VIA2. Jediglich die Signale für Write Protect und Track0 werden auch an den WD 1770 geleitet. Dadurch wird beispielsweise das Beschreiben geschützter Disketten verhindert, auch wenn die Betriebssoftware den Write Protect nicht prüfen würde.

Selbstverständlich wäre es mit einigen kleinen Basteleien möglich, den Steppermotor durch den WD 1770 ansteuern. Wenn man das Betriebssystem entsprechend ändert hätte das den Vorteil, daß das DOS nicht mehr beschäftigt ist und während der Kopf positioniert wird andere Aufgaben erledigen kann.

Eine weitere kleine Bastelanwendung ist, Pin 26 aufzutrennen. Dieses Signal ist auf Masse gelegt und es wird somit immer mit dem MFM-Aufzeichnungsverfahren gearbeitet. Verwenden Sie nun Disketten, die mit Single Density, also dem FM-Verfahren, beschrieben sind, können Sie den Controller darauf einstellen, indem Sie Pin 26 an 5V legen (möglichst mit 1KOhm-Widerstand). Sie können sich auch einen Umschalter einbauen, mit dem Sie Single und Double Density wechseln können (je nachdem ob an Pin 26 der Wert '0' oder '1' anliegt). Die Betriebssoftware der C-1570/71 merkt davon nichts und selbst das Betriebssystem CP/M+ arbeitet problemlos mit Single Density Disketten.

5.2.4 Die Technik des COMMODORE-Controllers

Bei COMMODORE-Formaten wird die Aufzeichnung über einen weiteren Controller gesteuert. Das Wort Controller ist allerdings etwas hoch gegriffen. Genau genommen handelt es sich nur um eine digitale Logikschaltung, an die ein gesamtes Byte übergeben wird, das diese dann seriell auf die Diskette schreibt. Diese Schaltung ist in einem Gate Array untergebracht, das COMMODORE in eigener Produktion fertigt. Aus diesem Grund fehlt hier die übliche Beschreibung der Pin-Belegung, da COMMODORE keine Unterlagen über diesen Baustein zur Verfügung stellt. Wer sich trotzdem für die Innereien des Gate Arrays interessiert, sollte sich mit den älteren Floppy-Modellen von COMMODORE beschäftigen. Denn das Gate Array wird

nicht verwendet, damit man die Schaltung nicht nachbauen oder analysieren kann, es ist vielmehr billiger als eine handvoll normaler diskreter Bauelemete.

Der Aufbau der COMMODORE-Hardware hat sich im Laufe der Jahre nur unwesentlich geändert. Aus diesem Grund kann besonders die Lektüre der CBM 4040 und der VC-1540 Unterlagen empfohlen werden, da diese Geräte die selben Grundfunktionen enthalten, wie die C-1570/71 - nur eben nicht in einem Gate Array.

Das Gate Array besteht aus zwei wesentlichen Baugruppen: einem Parallel-Seriell-/Seriell-Parallel-Wandler und einem BCD Zähler. Das zu schreibende Datenbyte wird von PA des VIA2 in ein Schieberegister übernommen. Von dort aus wird es mit dem Takt CLK, der durch einen programmierbaren Teiler und die Signale DS0 und DS1 (siehe VIA2) erzeugt wird, an die Schreibelektronik weitergegeben, die die Impulse verstärkt und die Kopfspule ansteuert.

Beim Lesevorgang geschieht dasselbe, nur in umgekehrter Reihenfolge. Darüber hinaus wird mit dem Eintreffen jedes '0'-Bits ein Zähler zurückgesetzt. Dieser Zähler beginnt beim Eintreffen von Bits mit dem Wert '1' zu arbeiten. Das geschieht solange, bis wieder ein '0'-Bit auftritt. Hat der Zähler den Wert 10 erreicht, dann wird das SYNC Signal ausgelöst, da bei normalen GCR-Daten nie mehr als 6 Einsen aufeinander folgen.

5.2.5 Der 1541 und 1570/71 Modus

Die Floppy kann in zwei verschiedenen Modi betrieben werden. Das ist zum einen der 1541-Modus und zum anderen der C-1570/71 Modus. Im 1541-Modus soll die Floppy zur VC-1541 kompatibel sein. Aus diesem Grund wird Sie in diesem Modus mit einer Taktfrequenz von 1MHz betrieben. Das DOS benützt dabei hauptsächlich die ROM-Routinen $C000 bis $FFFF, die mit dem

VC-1541 ROM identisch sind. Jediglich die Diskcontroller-Routine wird nicht benützt, da die neuen technischen Einrichtungen wie beispielsweise die Track0-Kennung nur vom 1571-ROM erkannt werden.

Ein weiterer gravierender Unterschied ist, daß Disketten nur einseitig beschrieben werden können, auch wenn es sich um beidseitig formatierte C-1571 Disketten handelt. Dieser Modus hat allerdings den Vorteil, daß die in 5.1.2 besprochenen Probleme mit beidseitigen Disketten, die im "Wendeverfahren" formatiert sind, nicht mehr auftreten.

Jedoch können im 1541-Modus die Zusatzfunktionen über User-0 nicht ausgeführt werden. Die einzige Ausnahme sind die Head- und Mode-Befehle, die teilweise im 1570/71-Modus nicht funktionieren, da sie da nicht erlaubt sind.

Der Bus kann im 1541-Modus nur den normalen C-64 Algorithmus bedienen Im 1570/71-Modus sind sowohl der alte Busmodus, als auch der neue, schnelle Busmodus möglich. Dies hängt vom Gerät ab, mit dem die Floppy kommuniziert und wird bei jeder Übertragung neu festgelegt.

Nach dem Einschalten ist die Floppy immer im 1541-Modus. Der 1571-Modus kann nur durch Umschalten mit dem 'u0>m1'-Befehl erreicht werden. Dies führt der C-128 immer beim Reset-Vorgang durch.

5.2.6 Der serielle Bus - Technik und Funktion

Der serielle Bus stellt die Verbindung zwischem dem Computer und den angeschlossenen Peripheriegeräten her. Über diese Übertragungsstrecke wird die Floppy angesteuert und werden Daten übertragen. Dieser Bus wird in diesem Kapitel sehr ausführlich besprochen. Auf diese Weise erfährt nicht nur der Einsteiger, wie die Datenkommunikation zwischen Floppy und Computer abläuft, sondern erhält auch der Profi handfeste Informationen für den Umgang mit dem Bus.

Der Bus besteht aus sechs Leitungen, deren Bedeutung nun detaillierter besprochen wird:

Pin	Name	Funktion
1	SRQ	Serial Request. Diese Leitung dient als Taktleitung beim schnellen Busmodus.
2	GND	Masseanschluß. Stellt zwischen allen angeschlossenen Geräten ein gemeinsames Null-Potential her.
3	ATN	Attention-Signal. Identifiziert Controller Befehle.
4	CLK	Taktleitung im normalen Busmodus.
5	Data	Datenleitung im normalen Busmodus.
6	Reset	Reset-Signal des Computers. Wird der Computer zurückgesetzt oder eingeschalten, so werden über diese Leitung auch alle angeschlossenen Periheriegeräte zurückgesetzt.

Die Leitungen GND (Pin2) und Reset (Pin6) können vom Computersystem nicht beeinflußt werden. GND ist das Massepotential des Computers. Über diese Leitung der Masseanschluß aller Peripheriegeräte verbunden, womit ein einheitliches 0V-Potential hergestellt wird, da sich Differenzen ausgleichen würden. Auf diese Weise entspricht der logische Pegel '0' bei allen Geräten der gleichen Spannung.

Die Reset-Leitung gibt den Impuls, der beim Einschalten oder durch den Reset-Knopf entsteht, an die Peripehriegeräte weiter. Diese verhalten sich dann so, als hätten sie den Reset-Impuls, der beim Einschalten des Peripheriegeräts entsteht, erzeugt und werden in den Ausgangszustand versetzt. Wird ein an den Computer angeschlossenes Gerät angeschaltet, darf kein Impuls auf der Reset-Leitung entstehen. Es gibt allerdings hin und wieder mangelhaft konstruierte Geräte, die dieses Verhalten aufweisen. Das hätte dann zur Folge, daß der Computer zurückgesetzt wird, wenn Sie ein angeschlossenes Gerät dazuschalten. Zuletzt ein Tip: schalten Sie zuerst immer alle benötigten Peripheriegeräte ein und danach den Computer. Dadurch können Sie sichergehen, daß alle Geräte im ordnungsgemäßen Ausgangszustand sind.

Die restlichen Leitungen des seriellen Busses werden vom Computer gesteuert. Der Computer ist dabei der sogenannte Controller, der die Aktion auf dem seriellen Bus steuert. Er legt fest, welche Geräte miteinander kommunizieren, welches Daten sendet und welches die Daten empfängt. Das diese Kommunikation meistens zwischen Computer und angeschlossenem Peripheriegrät stattfindet, ist zwar der Normalfall, aber keine zwingende Festlegung. Es können durchaus auch einmal zwei Floppy-Laufwerke miteinander Daten austauschen - die Buslogik wird dadurch nicht durcheinander gebracht. Nur werden derartige Übertragungen vom COMMODORE-Betriebssystem nicht unterstüzt.

Beim seriellen Bus muß beachtet werden, daß nur ein Gerät Daten senden darf, wobei dieses Talker genannt wird. Zuhören dürfen auch mehrere Geräte gleichzeitig, die Listener genannt werden. Welches Gerät als Talker und welches als Listener fungiert legt der Controller fest.

Nun werden Sie sicherlich verstehen, aus welchem Grund Sie zwei Computer nicht mit dem selben Peripheriegerät verbinden dürfen. In diesem Fall wären zwei Controller vorhanden, die beide gleichzeitig Anweisungen erteilen könnten, wer mit wem kommunizieren darf - ein heilloses Chaos wäre die Folge. Darüber gibt es eine Leitung, die nur von einem Controller gesteuert werden darf.

Das Problem, daß zwei Computer nicht verbunden werden dürfen, liegt nicht an der Buslogik, sondern an der Software, dem Betriebssystem. Sie können selbstverständlich ein Programm entwerfen, daß den Computer wie ein Peripheriegerät erscheinen läßt, welches nun an ein weiteres Computersystem angeschlossen werden kann.

Zur Übertragung der Daten stehen insgesamt vier Leitungen zur Verfügung, deren Funktion nun eingehender behandelt wird. da die Vorgänge auf dem Bus teilweise sehr kompliziert sind, werden wir die Methode der "schrittweisen Verfeinerung" anwenden. Das bedeutet, daß Sie zuerst die groben, grundlegenden Zusammenhänge kennenlernen. Später werden dann die einzelnen Grobabläufe detaillierter untersucht - bis zur Analyse des Schlatplans der Buselektronik.

Die zentrale Steuerung der Busvorgänge übernimmt der Controller. Dazu ist eine spezielle Leitung, ATN (Attention engl.= Achtung, Vorsicht) genannt, vorhanden, die vom Controller aktiviert wird, wenn er an die angeschalteten Peripheriegeräte ein Kommando richten möchte. Alle Daten, die bei aktiver ATN-Leitung (High-Zustand) gesendet werden, sind Kommandos. Diese bestehen normalerweise aus zwei Bytes, wobei das erste Byte folgendermaßen aufgebaut ist:

```
Bit0-4: Geräteadresse des angesprochenen Geräts.
Bit5:   1= Adressierung als Listener.
Bit6:   1= Adressierung als Talker.
Bit7:   0= Kennzeichen für Primäradresse (1.Befehlsbyte).
        1= Kennzeichen für Sekundäradresse.
```

Bit 7 der Kommandobytes zeigt an, ob es sich um das erste Byte, die Primäradresse, oder das zweite Byte, die Sekundäradresse handelt. Beim Aufruf eines Gerätes wird zuerst das 1. Kommandobyte gesendet. Die Bits 5 und 6 geben dabei an, ob das Gerät als Talker oder als Listener agieren soll. Durch diese beiden Bits wird also gesteuert, ob das angesprochene Gerät Daten senden oder Daten empfangen oder senden soll. Es darf immer nur eines der beiden Bits gesetzt sein, da ein Gerät nicht gleichzeitig Daten senden und empfangen kann.

Die Bits 0-4 des Primärbytes geben die Nummer des Geräts an, für welches das Kommando bestimmt ist, die zwischen 0 und 30 liegen darf. Aus diesem Grund muß jedes Gerät, immer wenn die ATN-Leitung im aktiven Zustand ist, die Vorgänge auf den Bus mitverfolgen um zu erkennen, wenn es angesprochen wird.

Die Geräteadresse 31 hat eine Sonderfunktion und dient dazu den Datenverkehr auf dem Bus wieder einzustellen. Wird diese Adresse angesprochen, dann setzen alle Peripheriegeräte ihre Buslogik zurück und beenden die aktuelle Talker- oder Listener-Funktion. Üblicherweise ist davon nur ein Gerät, nämlich das, daß mit dem Computer kommuniziert, betroffen. Doch es gibt auch die Möglichkeit, daß zwischen zwei Peripheriegeräten Daten ausgetauscht werden oder mehrere Geräte gleichzeitig im Listener-Betrieb sind. In diesem Fall werden durch die Geräteadresse alle Geräte gleichzeitig zurückgesetzt. Bei dieser Rucksetz-Funktion haben die Bits 5 und 6 wieder die gleiche Beduetung wie bei den normalen Geräteadressen. Auf diese Weise können Listener und Talker getrennt zurückgesetzt werden, wobei diese Buskommandos dann die Bezeichnungen Unlisten oder Untalk erhalten.

In den meisten Fällen reicht die Adressierung des Peripheriegeräts nicht aus, da man spezielle Funktionen des Geräts ansteuern möchte, was durch die Sekundäradresse bewirkt wird. Soll eine Sekundäradresse angesprochen werden, dann wird im ATN-Kommandomodus ein weiteres, zweites Steuerbyte gesendet. Dieses kann amm gesetzten Bit 7 erkannt werden. Die Bits 5 und 6 haben beim Sekundärbyte immer den Wert '1'. Bit4 gibt an, ob der Sekundärkanal geöffnet oder geschloßen werden soll. Die Nummer des gewünschten Kanals muß in Bit 0-3 angegeben werden.

Nun nochmals die verschiedenen Steuerbytes im Überblick:

```
Steuerbyte   Funktion
--------------------------------------------
010xxxxx     Talker-Aufruf.
01011111     Untalk.
001xxxxx     Listener-Aufruf.
00111111     Unlisten.
--------------------------------------------
1111yyyy     Sekundäradresse eröffnen.
1110yyyy     Sekundäradresse schließen.
0110zzzz     Sekundäradresse für Listener- und
             Talker-Betrieb übermitteln.
--------------------------------------------
```

Wie Sie schon wissen, muß im ATN-Modus jedes Peripheriegerät die Kommandobytes analysieren um festzustellen, ob es von dem Befehl betroffen ist. Was passiert nun, wenn ein Gerät gerade beschäftigt ist? Stellen Sie sich vor, die Floppy formatiert eine Diskette oder der Drucker gibt gerade eine Zeile aus. In dieser Phase ist das Prozessorsystem des Periepheriegerätes umfassend mit der auszuführenden Arbeit beschäftigt und hat keine Zeit, den Busverkehr zu beobachten.

Aus diesem Grund wurde ein Verfahren, daß 'Handshaking' genannt wird, entwickelt, mit dessen Hilfe der Datenfluß gesteuert wird. Genauso wie Sie nicht einfach in das Büro Ihres Chefs hineinstürmen, sondern erst einmal anklopfen und auf das 'Herein' warten, werden die Bytes nicht sofort auf den Bus ausgegeben, sondern erst einmal festgestellt, ob alle Geräte zuhören.

Die dazu nötigen Steuersignale werden über die Clock- und die Daten-Leitung gesendet und haben folgende Bedeutung:

```
Data  0= alle Peripheriegeräte bereit.
      1= Peripheriegerät(e) nicht bereit.
Clock 0= Controller bereit.
      1= Controller sendet noch keine Daten.
```

Soll ein Gerät adressiert werden geschieht folgendes:

Zuerst wird die ATN-Leitung aktiviert, womit signalisiert wird,, daß nun ein Kommando folgt. Darauf setzt der Controller die Clock-Leitung auf den Wert '1' um mitzuteilen, daß das Datenbyte noch nicht übertragen wird. Gleichzeitig sorgt er dafür, daß auf der Data-Leitung der Wert '0', also kein Signal, anliegt. Durch eine elektronische Schaltung wird an jedem Peripheriegerät automatisch die Daten-Leitung auf den Zustand '1' gesetzt.

Nun haben die Peripheriegeräte Zeit, sich auf den Empfang der Steuerbytes vorzubereiten. Ist das Peripheriegerät bereit das Steuerbyte zu verarbeiten, dann setzt es seinen Daten-Ausgang wieder auf den Wert '0'. Hat das letzte Peripheriegerät die Datenleitung zurückgesetzt, geht der Pegel der Datenleitung auf den Wert '0' zurück, wodurch dem Controller signalisiert wird, daß alle Geräte bereit sind.

Ob überhaupt ein Gerät angeschlossen ist, wird vom Controller zu Anfang dieser Adressierungphase festgestellt. Dazu prüft er, ob die Daten-Leitung innerhalb einer Millisekunde auf den Wert '1' gestetzt wird. So hat auch ein Peripheriegerät, das keine automatische Schaltung zum setzen der Daten-Leitung hat, genügend Zeit, auf den Controller-Aufruf zu reagieren. Bleibt das Hochsetzen der Daten-Leitung aus, dann gibt der Computer die Fehlermeldung 'Device not present' aus.

Sind alle Peripheriegeräte bereit, dann setzt der Controller die Clock-Leitung auf den Zustand '0'. Dies ist für die angeschlossenen Geräte das Signal, daß nun die Übertragung beginnt. Nun erfolgt kein Handshaking mehr. Die Peripheriegeräte müssen nun alle schnell genug sein, um die Daten zu verarbeiten. Mit jedem Low-Impuls auf der Clock-Leitung wird ein Datenbit auf der Daten-Leitung ausgegeben. In der Zeitspanne zwischen zwei Datenbits nimmt die Clock-Leitung den Zustand '1' an, wodurch den angeschlossenen Geräten mitgeteilt wird, daß sie sich nun melden sollen, ob sie das Datenbit emfangen haben. dazu setzt der Talker, der in diesem Fall der Controller ist, die Daten-Leitung auf den Wert '0'. Hat das Peripheriegerät das Datenbit empfangen, muß es dies dem Talker mitteilen, indem es auf der Daten-Leitung einen High-Pegel ausgibt. Dabei beachtet der

Talker immer nur das erste Gerät, das sich auf diese Weise meldet. Sind weitere Zuhörer vorhanden, dann werden die Meldungen dieser Geräte ignoriert. Für den Talker ist es nur interessant ob überhaupt ein Listener vorhanden ist. Erfolgt die Quittierung des Datenbits nicht, dann erfolgt eine 'Time out'-Fehlermeldung im Statusbyte des Computers,.

Die grundlegenden Vorgänge auf dem seriellen Bus sind im Grunde genommen nicht sehr kompliziert, trotzdem können Sie nun, besonders in 6502-Maschinensprache, noch keine eigenen Bussteuerroutinen entwerfen. Dazu müssen Sie nämlich auch die Hardware des seriellen Busses kennenlernen.

Wenn Sie allerdings die Routinen des Betriebssystems verwenden, ist die Busprogrammierung nicht schwer. Es ist für jede Busfunktion, wie Talk oder Untalk, eine eigene Routine vorhanden, die in Maschinensprache nur aufrufen müssen. Dabei müssen in einigen Zeropagestellen und den Prozessor-Registern, die Parameter für den Busaufruf übergeben werden.

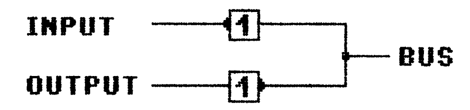
Hier nun eine Zusammenstellung der wichtigsten Betriebssystem-Routinen, die auf dem VC-20, C-64 und C-128 identisch sind:

```
Name      Adresse   Funktion/Parameter
-----------------------------------------------------
Parameter in Zeropageadressen, die immer gesetzt sein
müssen:

$B8       Logische Dateinummer
$BA       Geräteadresse
$B9       Sekundäradresse + Steuerbits 4-7
$BB/BC    Adresse des Dateinamens bei Open
$B7       Länge des Dateinamens
-----------------------------------------------------
OPEN      $FFC0     Eröffnet Datenkanal (wie in BASIC)
CLOSE     $FFC3     Schließt Datenkanal (wie in BASIC)
CKOUT     $FFC9     Gibt Zeichen aus A auf Bus aus
CHKIN     $FFC6     Holt Zeichen vom Bus nach A
TALK      $FFB4     ruft Talker-Funktion auf
LISTEN    $FFB1     ruft Listener-Funktion auf
SECTALK   $FF96     Sendet Sekundäradresse nach Talk
```

```
SECLISTEN $FF93    Sendet Sekundäradresse nach Listen
UNTALK    $FFAB    Sendet Untalk-Kommando
UNLISTEN  $FFAE    Sendet Unlisten-Kommando
--------------------------------------------------------
```

Die Assembler-Profis unter Ihnen möchten selbstverständlich nichts dem Betriebssystem überlassen, sondern alles selbst programmieren. Sicherlich haben Sie deshalb schon die Belegung der Ein-/Ausgabe-Bausteine der Floppy und des Computers studiert.



Dabei fällt auf, daß für ein und die selbe Busleitung zwei Verschiedene Bits des Ein-/Ausgabeports verwendet werden. Das eine Bit dient zur Ausgabe der Daten und das andere Bit ist als Eingang geschaltet. Um zu verstehen, was hinter dieser merkwürdigen Belegung steckt, müssen Sie den Schaltplan der Buslogik zu Rate ziehen.
Wie Sie sehen, werden die beiden Anschlüße des Ein-/Ausgabe-Bausteins über zwei Inverter mit der Busleitung verbunden. Der Inverter gibt immer genau den entgegengesetzten Pegel aus, wie am Eingang des Inverters anliegt. Auf diese Weise haben die Pegel auf den Leitungen des seriellen Busses immer den entgegengesetzten Wert wie das Bit im Ein-/Ausgabe-Register.

Der Grund dafür ist elektrotechnischer Natur. Es ist nämlich physikalisch nicht möglich auf eine Leitung den '0'-Pegel zu setzen und dann die Leitung durch einen anderen Ausgang, beispielsweise am Peripheriegerät, mit dem Wert '1' ansteuern. Die Leitung würde nie den Zustand '1' annehmen, da der Null-Pegel wie ein Kurzschluß wirkt.

Also muß die ganze Handhabung der Signale invertiert werden. Soll die Leitung den Wert '0' haben, wird nun der Pegel '1' ausgegeben. Dieser liegt solange an, bis an irgend einem ange-

schlossenen Gerät der Pegel '0' gesetzt wird. Dadurch entsteht ein Kurzschluß und die Spannung auf der Leitung bricht zusammen. Dies entspricht dann dem Null-Pegel.

Die Geräte, die am seriellen Bus angeschlossen sind, haben nun immer eine Ausgangsleitung und eine Eingangsleitung. Durch die Ausgangsleitung wird ein bestimmter Pegel am Bus angelegt. Wenn es sich dabei um einen '1' Pegel handelt, also der Wert '0' auf den Bus gegeben wird (beachte Inverter), kann es durchaus vorkommen, das dieser Spannungspegel von einem anderen angeschlossenen Gerät auf den Zustand '0' gebracht wird. Aus diesem Grund hat jedes Gerät eine Eingangsleitung, mit der der wirklich anliegende Buspegel abgefragt werden kann.

Im Grunde genommen ist diese spezielle Hardware-Organisation für die Programmierung nicht interessant, da die Elektronik durch die Inverter immer wieder die richtigen logischen Werte herstellt. Doch es gibt leider eine Ausnahme. Dies ist der Eingang am Rechner, dem C-64 oder dem C-128. Dieser Eingang hat keinen Inverter vorgeschaltet.

Das muß bei der Programmierung beachtet werden. Sollen die Werte der Buseingänge am Rechner abgefragt werden, müssen diese immer durch das Programm invertiert werden. Soll gewartet werden, bis ein bestimmter Pegel an einer Busleitung anliegt, dann müssen Sie darauf achtgeben, daß Sie auf den entgegengesetzten Wert warten müssen.

Der Sachverhalt, daß die physikalischen Pegel im Gegensatz zu den logischen Werten invertiert sind, führt besonders in ROM-Listings immer wieder zu Verwirrungen. Suchen Sie deshalb immer zuerst die Routinen, die die Daten- oder die Clock-Leitung auf einen bestimmten Wert setzen. Daran können Sie erkennen ob das Setzen einer Leitung auf den Wert '1', mit dem physikalischen oder mit dem logischen Pegel kommentiert wird. Im DOS-Listing im Floppy-Buch werden immer die logischen Pegel kommentiert. Besondere Vorsicht ist bei ROM-Listings zum C-64 oder C-128 geboten. Darin wird nämlich meistens übersehen, daß die Eingänge nicht invertiert werden und somit die physikalischen Buspegel anliegen. Dies führt dann zu einem

munteren Kommentar-Wirrwarr, daß mit den eigentlichen Vorgängen am seriellen Bus nichts mehr zu tun hat.

Wie Sie wissen kann die C-1570/71 im schnellen Busmodus betrieben werden. Zu diesem Modus wird die SRQ-Leitung benötigt. Über diese Leitung werden die Taktsignale übertragen, die das CIA 6526 bei der Ausgabe des Datenbytes erzeugt. Mit diesen Taktsignalen wird das Datenregister des empfangenden Geräts gesteuert.

Über die SRQ-Leitung wird auch festgestellt, ob das angesprochende Peripheriegerät im schnellen Busmodus angesprochen werden kann. Dazu sendet das aufgerufene Gerät beim Zurücksetzen des Pegels auf der Daten-Leitung, womit es seine Emfangsbereitschaft signalisiert, acht Taktimpulse auf der SQR-Leitung. Diese Impulse setzen ein Flag im Interruptregister des Computers. Wenn der Controller das Zurücksetzen der Daten-Leitung erlkennt, prüft er nun gleichzeitig, ob die das Flag im Interruptregister gesetzt ist. Somit weiß der Bus-Controller, daß das angeschlossene Gerät im schnellen Busmodus betrieben werden kann. In diesem Fall sendet der Controller beim Setzen des Clock-Signals, ebenfalls acht Taktimpulse über die SRQ-Leitung. Dadurch wird dem Peripheriegerät mitgeteilt, daß es die Daten im schnellen Busmodus senden oder empfangen soll.

Beim Schnellmodus werden die Datenbits und die Taktbits nicht invertiert, da dies das CIA nicht verarbeiten würde. Nur die eigentliche Datenübertragung wird im schnellen Busmodus durchgeführt. Dabei wird ein Byte in genau 64 Mikrosekunden übertragen, wobei eine Übertragungsrate von 15625 Bytes pro Sekunde möglich wäre.

Wie Sie aber schon bemerkt haben, werden im Schnell-Modus gerade 3500 Bytes geladen. Dies liegt an den Verwaltungsroutinen der Floppy und des Computers, die beide nicht besonders ausgeklügelt sind und deshalb die Übertragung bremsen.

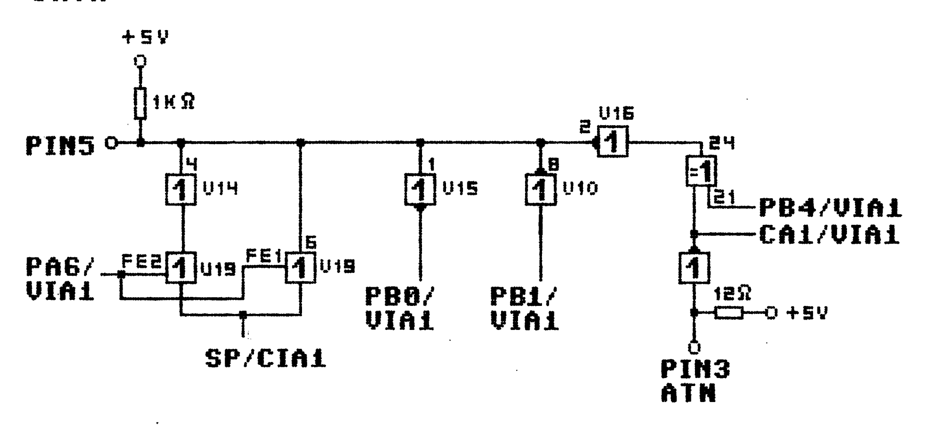
Dies war auch schon beim C-64 der Fall. Der Übertragungsalgorithmus des C-64 Bus ist durchaus fähig bis zu 1200 Bytes pro Sekunde zu senden oder zu empfangen. Die umständlichen Ver-

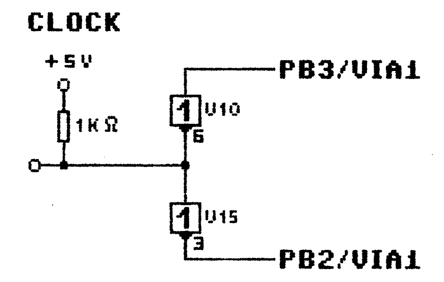
waltungsroutinen der Betriebssysteme bremsen die Busgeschwindigkeit auf magere 400 Bytes pro Sekunde.

Aus diesem Grund ist der C-128 beim Speichern von Programmen nicht schneller als der C-64, obwohl der schnelle Bus verwendet wird. Denn es ist unwichtig, ob die Daten zum Übertragen eine Millisekunde oder einige Mikrosekunden benötigen, wenn das Betriebssystem nur alle 2.5 Millisekunden ein Byte übernehmen kann.

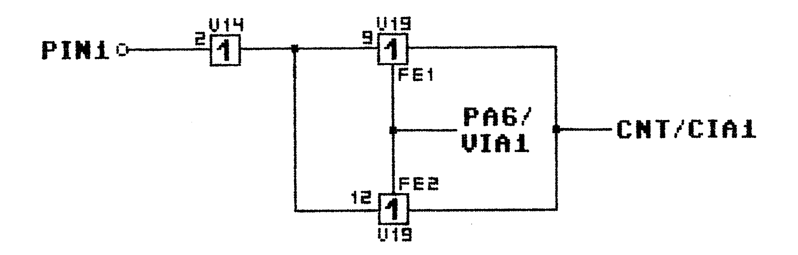
Die Schnell-Lade-Systeme sind vorwiegen nur deshalb schneller, weil die Verwaltungsarbeit (Datei eröffenen, Pointer verwalten...) drastisch gekürzt wurde.

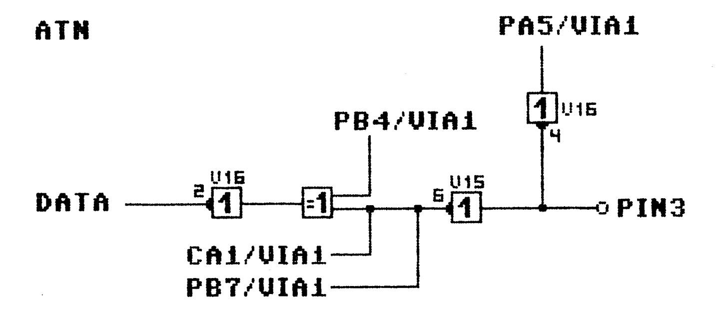
Theoretisch ist es sogar möglich mit der Technik des C-128 Fast-Bus Übetragungsraten mit bis zu über 60000 Bytes pro Sekunde zu realisieren.


DATA
+5V
1KΩ
PIN5
4
U14
PA6/
VIA1
FE2
U19
FE1
6
U19
SP/CIA1
1
U15
PB0/
VIA1
8
U10
PB1/
VIA1
2
U16
24
21
PB4/VIA1
CA1/VIA1
12Ω
+5V
PIN3
ATN


CLOCK
+5V
1KΩ
PB3/VIA1
U10
6
U15
3
PB2/VIA1

SQR


PIN1
U14
2
1
U19
9
1
FE1
PA6/
VIA1
CNT/CIA1
FE2
12
1
U19


ATN
PA5/VIA1
1
U16
4
PB4/VIA1
DATA
U16
2
1
=1
U15
6
1
PIN3
CA1/VIA1
PB7/VIA1

5.2.7 Der Steppermotor

Der Steppermotor ist eine besondere Antriebseinheit. Ein normalen Motor beginnt zu laufen, sobald er mit Strom versorgt wird. Ein derartiger Motor ist zum Beipiel der Laufwerksmotor, der die Diskette in Rotation versetzt.

Dieser Laufwerksmotor hat auf seiner Achse einen Tachogenator, eine kleine Komponente, die die Umdrehungszahl des Motors feststellt. Aufgrund dieser Messung wird die Stromzufuhr des Laufwerksmotors gesteuert, sodaß der Motor konstant läuft.

Der Stepper dagegen ist ein Spezialmotor, der nicht mit einem gleichmäßigen Strom sondern mit Impulsen angesteuert wird. Bei jedem Impuls bewegt sich der Motor um einen ganz bestimmten Winkel. Beim Steppermotor der C-1570/71 sind das geanau 1.8 Grad. Das bedeutet, daß der Motor nach genau 200 Impulsen eine Umdrehung vollzogen hat.

Den Drehwinkel eines Steppermotors nennt man auch Schritt oder Step, womit nun auch die Namensgebung verständlicher wird. Bei jedem Impuls bewegt sich der Motor um einen Schritt. Der Vorteil des Steppermotors ist, daß er sowohl vorwärts als auch rückwärts gedreht werden kann. Die Drehrichtung wird bei der C-1570/71 durch die Signale STP0 und STP1. Diese beiden Bits müssen als 2-Bit-Wert betrachtet werden. Erhöht man den Wert, dann bewegt sich der Motor so, daß der Kopf nach außen fährt, wird dieser Wert erniedrigt, so bewegt sich der Kopf nach innen.

Bei der C-1570/71-Floppy wird der Kopf durch zwei Schritte um eine Spur weiterbewegt. Diese Schrittimpule dürfen selbstverständlich nicht zu schnell erfolgen, denn der Steppermotor benötigt eine gewisse Zeit, bis er den Schritt ausgeführt hat. Wie Sie sicher schon bemerkt haben bewegt sich der Kopf im 1571-Modus wesentlich schneller, als im 1541-Modus.

Dadurch, daß im 1571-Modus die Taktfrequenz doppelt so hoch ist, werden die Schrittimpulse auch doppelt so schnell erzeugt. Allerding sollten Sie bei eigenen Experimenten bei der Stepperprogrammierung sehr behutsam vorgehen und am Besten die Betriebssystemroutinen verwenden.

KAPITEL 6
1571-PRAXIS

6.1 Spitzen-Diskmonitor

6.1.1 Die Befehle des Diskmonitors

Der Diskmonitor ist ein Diskettenwerkzeug, wie Sie es in anderen Publikationen nicht finden werden. Auch auf dem Markt der professionellen Programme werden Sie etwas Ähnliches vergeblich suchen. Das Listing des Diskmonitors ist allerdings so umfangreich, daß ein Abtippen sehr aufwendig wäre, zumal es fast 20 Seiten dieses Buches in Anspruch nehmen würde. Aus diesem Grund ist der Diskmonitor auf einer zum Buch erscheinenden Diskette enthalten.

Allgemeine Bedienungshinweise :

Der Diskmonitor hat einen eigenen Bildschirm-Editor. Dieser läßt den Cursor nur in Bildschirmteile springen, die Daten enthalten. Verläßt man mit dem Cursor ein Datenfeld, so wird der in diesem Datenfeld eingetragene Wert übernommen. Mit der Returntaste springt der Cursor an den Anfang der nächsten Zeile. Ein Druck auf die 'Home'-Taste läßt den Cursor auf das erste Datenfeld springen. Drückt man die 'Home'-Taste zweimal, dann kommt man zum Kopf der Anzeige, wo die Parameter der Spur oder des Sektors eingegeben werden.

Das Programm ist in verschiedene Module unterteilt, die jeweils Sektoren, Spuren usw. bearbeiten. Diese Module werden über ein Menü angewählt. Die 'Stop'-Taste bewirkt den Rücksprung zum übergeordneten Menü. Mit 'Stop' und 'Restore' versetzt man den Diskmonitor in den Ausgangszustand. Laufende Operationen werden sofort abgebrochen.

Der Aufruf der einzelnen Befehle des Monitors erfolgt über die Tastatur. Zuerst muß die Control-Taste (CTRL) gedrückt werden. Dann ist die Anzeige CTRL in der Bildschirmzeile, in der alle Befehle aufgelistet sind, revers dargestellt. Nochmaliges Drücken der CTRL-Taste macht dies wieder rückgängig. Ist das Programm im CTRL-Modus, dann werden die Funktionen des Diskmonitors einfach durch Tastendruck aufgerufen.

Allgemeine Steuerfunktionen :

CTRL 1/2/3 :
Mit diesen Tasten koennen Schrift-, Hintergrund- und Rahmenfarbe des Bildschirms nach eigenem Geschmack eingestellt werden.

CTRL > :
Dies erlaubt die Eingabe der 1571/1541 DOS-Befehle. Diese Befehle werden dann vom DOS, ohne Eingriff oder Modifikation durch den Diskmonitor ausgefuehrt. Es ist immer wieder nuetzlich, um zu sehen, wie das DOS auf bestimmte Monitormanipulationen reagiert. Sollen Zeichen gesendet werden, die nicht über die Tastatur erreichbar sind, dann kann nach dem '^'-Zeichen der ASCII-Wert des Zeichens dezimal angegeben werden. Nach der Dezimalzahl muß unbedingt ein Leerzeichen folgen.

CTRL x :
Dieser Befehl fuehrt einen Rechner-Reset aus und kehrt zum Basic zurueck. Vorher wird mit 'Are you sure' nachgefragt, ob der Anwender dies auch wirklich moechte.

CTRL +/- :
Damit laesst sich der Kopf des Laufwerks um eine Halbspur nach außen oder nach innen bewegen. Dies ist bei dejustierten Laufwerken oder Disketten, die auf einem derartigen Laufwerk beschrieben wurden, besonders wichtig. Dieser Befehl hat eine Autorepeat-Funktion, wenn die Taste laenger gedrueckt wird. Hat sich der Kopf auf eine unformatierte Spur 'verirrt' und findet nicht zurück, kann er mit diesem Befehl 'von Hand' gesteuert werden.

CTRL 0 :
Setzt den Kopf auf Spur Null zurueck.

CTRL 8/9 :
Waehlt Laufwerk 8 oder 9 als aktives Laufwerk. Welches Laufwerk ausgewaehlt ist, wird meistens in der CTRL-Zeile mit 'D:8' angezeigt.

Wenn Sie den Diskmonitor starten, erscheint zuerst das Hauptmenü mit folgenden Menüpunkten :

DIRECTORY
BAM
TRACK
SECTOR
MEMORY
CP/M

Der Aufruf CP/M führt zu einem ähnlichen Menü, nur daß CP/M Disketten bearbeitet werden. Die anderen Menüpunkte rufen das entsprechende Modul auf. Diese werden jetzt besprochen. Beginnen wir mit dem

DIRECTORY MODUL

Sie sehen nun folgendes Bildschirmbild (Ausschnitt) :

```
-----------------------------------------
CTRL : a b l n r s t v w = > *       D:8
--------------[DIRECTORY]----------------

NR  NAME              T  BL  TR SE

001 test1             p< 023 17 01
002 test2             s  010 19 03
                      .
                      .
```

In diesem Programmteil dreht sich alles um das Directory. Nun zu obigem Bildschirmausschnitt. Zuerst wird die Nummer des Directoryeintrags angezeigt (NR). Danach folgt der Name der Datei. Sind im Namen Steuerzeichen enthalten, die im normalen Directory nicht sichtbar sind, werden diese invers dargestellt. Genauso werden auch invers eingegebene Zeichen nachher in Steuerzeichen umgewandelt. Die Spalte 'T' gibt den Dateityp an, wobei p/s/r/u/d für PRG/SEQ/REL/USR/DEL stehen. Rechts von der Spalte werden noch die Bits 6 und 7 der Dateitypen

angezeigt. Dabei ist die Bedeutung von '*' und '<' mit dem normalen Directory identisch.

Nun folgen die Spalten TR und SE, die für Track (Spur) und Sektor stehen. Es handelt sich dabei um Spur- und Sektornummer des ersten Datenblocks des Directory-Eintrags. Kommt man mit dem Cursor an den rechten Rand der Spalte SE und betätigt nochmal 'Cursor rechts', dann wird auf eine weitere Anzeige umgeschaltet. Es erscheint folgendes Bildschirmbild (Ausschnitt) :

```
NR  NAME             T  START END

001 test1            p  $0801 $4c23
002 test2            s
003 test3            r
                     .
                     .
```

Bevor diese Spalten erscheinen, läuft die Floppy an und der Diskmonitor liest die Sektoren aller Directoryeinträge. So wird bei Programmen die Start- und Endadresse ermittelt. Diese Daten dürfen auch geändert werden. Schreibt man das Directory wieder auf die Diskette zurück, dann werden auch alle geänderten Startadressen in die ersten Sektoren der Programmdateien eingetragen. Natürlich gibt eine Start- und Endadresse bei Dateien keinen Sinn. Deshalb erfolgt hier keine Angabe.

Aber auch zu den Dateien liefert der Diskmonitor weitere Informationen. Dazu muß man wieder mit dem Cursor über den rechten Rand hinausfahren. Dann erhält man untenstehende Anzeige. Zuerst werden Spur (TR) und Sektor (SE) des ersten Side-Sektorblocks genannt. Darauf die Länge eines Datensatzes (LE). Diese Angaben sind natürlich nur bei relativen Dateien möglich. In der Spalte REC ist die Zahl der in der Datei enthaltenen Datensätze (Records) aufgeführt. Diese müssen immer durch CHR$(13) voneinander getrennt sein. Die Zahl dient nur zur Information und kann nicht geändert werden. Das ist nur durch entsprechende Dateiverwaltungsprogramme, wie sie in den Kapiteln 1.4 und 1.5 besprochen werden, möglich.

Die Bildschirmanzeige für Dateien :

```
NR  NAME          T  TR SE LE  REC

001 test1         p
002 test2         s            01260
003 test3         r  19 10 120 00370
                  .
                  .
                  .
```

Nun die CTRL-Befehle des Directorymoduls :

CTRL a (alphasort):
Sortiert alle Einträge eines Bereichs oder des ganzen Directories alphabetisch. Der Dateityp kann dabei als Sekundärmerkmal beachtet werden.

CTRL b (blanks/noblanks):
Entfernt bei allen Einträgen des Directories Leerzeichen, die nach dem Dateinamen stehen, oder fuellt alle Eintraege mit Leerzeichen auf.

CTRL l (links):
Zeigt alle Verkettungszeiger (Links) der Datei an, in dessen Zeile der Cursor steht.

CTRL n (new):
Rettet Disketten nach 'new'. Dabei werden die Linkbytes jedes Sektors in den Rechner eingelesen (Dauer etwa 20 sec.). Danach sucht der Monitor alle Sektoren die als Link '00' enthalten. Dies Sektoren stellen den letzten Sektor einer Sektorverkettung dar. Im weiteren werden die Sektoren, die auf diese Endsektoren zeigen gesucht. Auf diese Weise erstellt die Funktion aus den Verkettungen der Sektoren ein Directory und die BAM.

CTRL r (read):
Liest Directory in den Speicher ein.

CTRL s (sort):
Dateieintrage werden mit dem Cursor 'angeklickt' und dann an eine andere Stelle im Directory verschoben. Damit kann das Directory nach eigenen Wünschen sortiert werden.

CTRL t (trace):
Liest alle Dateieintrage, um die Zusatzinformationen der erweiterten Darstellung (siehe oben) zu gewinnen. Diese Informationen bleiben bis zum nächsten 'CTRL r' im Rechner gespeichert. Alle Zusatzinformationen (ausser REC) koennen editiert und mit 'CTRL w' auf Diskette geändert werden.

CTRL v (validate):
Liest alle Linkbytes der Diskette ein und erstellt dann, anhand des Directory, eine neue BAM. Der Gesamtvorgang dauert etwa 30 Sekunden - eine gewaltige Geschwindigkeitssteigerung gegenueber dem DOS-Validate (Collect).

CTRL w (write):
Schreibt alle Daten zurück. Zuvor wird zur Sicherheit der Benutzer gefragt, ob er das alte Directory und die BAM wirklich überschreiben will.

CTRL = :
Fügt einen Trennstrich in das Directory ein. Der Filetyp ist dabei DEL. Dies ermöglicht die übersichtliche Gestaltung des Inhaltsverzeichnisses.

CTRL > :
Setzt oder löscht Bit 6 im Dateitypbyte. Bit 6 kennzeichnet die Datei, daß sie nicht mehr gelöscht oder umbenannt werden kann.

CTRL * :
Setzt oder löscht Bit 7 im Dateitypbyte. Durch Bit 7 wird die Datei als ordnungsmässig geschlossen und lesbar oder als unlesbar markiert.

DEL / INST :
Eintrag löschen oder Leereintrag einfügen.
TRACK MODUL

Dieses Modul verarbeitet gesamte Spuren (Tracks). Dabei werden alle Bytes und Informationen, selbst die Sync-Markierungen, angezeigt. Die Daten werden immer ab dem Beginn des Indexlochs gelesen. Da die Spuren unterschiedliche Längen haben, wird das Einlesen einfach bis zum Indexloch durchgeführt. Die Darstellung der Spur kann in GCR-Code oder in Binär erfolgen. Wird die Binärdarstellung gewählt, dann sind alle Sync-Bytes durch 'SS' ersetzt. Wird statt eines Binärwerts an irgendeine Stelle 'SS' geschrieben, dann erzeugt der Diskmonitor beim Zurückschreiben der Spur an dieser Stelle eine Sync-Markierung. Neben der Hexadezimal-Darstellung sind auch die den Byte-Werten entsprechenden ASCII-Zeichen angegeben.

Nun die Bildschirmmaske des Track-Moduls:

```
----------------------------------------
CTRL : r w n i f s c l             D:8
---------------[TRACK]-----------------
GCR : 0  TRACK : 01  SIDE : 0
----------------------------------------
 0000 40 41 42 43 00 00 00 00  @abc....
 0008 00 00 00 00 31 39 38 35  ....1985
 0010 00 00 00 SS SS SS SS 08  ........
 0018 12 04 01 41 42 0F 0F 00  ...ab...
 0020 00 00 00 00 00 00 00 SS  ........
 0028 SS SS SS 07 01 02 03 04  ........
                  .
                  .
                  .
```

Das Datenbyte in der linken oberen Ecke wird mit 'Home' erreicht. Betätigt man die 'Home'-Taste zweimal, dann fährt der Cursor auf die Track-Angabe. Nochmaliges Betätigen von 'Home', also insgesamt dreimal, setzt den Cursor auf die Angabe 'GCR'-Modus. Eine 0 bedeutet, daß die Bytes der Spur als Binärwerte angezeigt werden. Die Ziffer 1 schaltet hierbei auf die GCR-Darstellung.

Befehle des Trackmoduls :

CTRL r (read): Liest die Spur in den Rechner ein.

CTRL w (read): Schreibt die Spur auf die Diskette zurück.

CTRL n (new):
Formatiert eine Spur neu. Damit können Disketten in Mischformaten (Commodore + CP/M) hergestellt werden. Es müßen unbedingt zwei ID Zeichen angegeben werden. Diese werden in die Sektorheader geschrieben und sollten auf der ganzen Diskette identisch sein. Ansonsten erfolgt ein 'Disk ID Mismatch' Fehler.

CTRL i (id-change):
Die ID einer Diskette ist nicht, wie immer noch gemeint wird, in Sektor 18/01 gespeichert. Diese ID-Zeichen dienen nur nur zur Anzeige im Directory. Das DOS orientiert sich an der ID, die in jedem Sektorheader gespeichert ist. Diese Funktion ändert die ID-Zeichen in jedem Sektorheader der Diskette oder eines Disketten-Bereichs.

CTRL f (fill):
Beschreibt die gesamten Spur oder einen Teil der Spur mit einem beliebigen Wert. Damit können zum Beispiel sogenannte Killertracks gelöscht oder hergestellt werden (Kopierschutz).

CTRL s (sectors):
Sektoren einer Spur oder eines Spurbereichs mit einem Wert füllen.

CTRL c (copy):
Kopiert einen Teil der Spur an eine andere Stelle auf der Spur.

CTRL l (location):
Damit kann man den physikalischen Beginn der Spur festlegen. Der Cursor muß auf dem Byte stehen, das nach dem Indexloch geschrieben werden soll.

SEKTOR MODUL

Vorweg eine besondere Eigenschaft, die das Sektormodul gegenueber anderen Diskmonitoren hat. Es gibt für den Monitor keine Lesefehler, außer er findet die Sync-Markierung und damit den Sektor nicht (dabei hilft eventuell das Trackmodul). Der Grund dafür liegt darin, daß alle Informationen eines Sektors angezeigt werden - auch die Prüfsummen usw.

```
----------------------------------------
CTRL : r w s f h c t z n p # $ " '    D:8
---------------[SECTOR]-----------------
DS: 0  TR: 24 SE: 10  POS: 000
----------------------------------------
 HE: 008 CKS: 021 TR: 24 SE: 10 ID: ab
 BL: 007 CKS: 122
----------------------------------------
 00  40 41 42 43 00 00 00 00  @abc....
 08  00 00 00 00 31 39 38 35  ....1985
 10  00 00 00 00 00 00 00 00  ........
                 .
                 .

----------------------------------------
 HE: 008 CKS: 021 TR: 24 SE: 10 ID: ab
 BL: 007 CKS: 122
----------------------------------------
 024  123 231 233 133 123 123 222 312
 032  123 231 233 133 123 123 222 312
 040  123 231 233 133 123 123 222 312
                 .
                 .
```

Die Darstellung des Sektorinhalts kann dezimal oder hexadezimal erfolgen. Ist die hexadezimale Anzeige ausgewählt, dann werden alle Bytes zusätzlich im ASCII-Code wiedergegeben. Anstatt der ASCII-Anzeige kann die Anzeige auch im Bildschirmcode erfolgen. Nicht druckbare Zeichen (Steuerzeichen) sind durch einen Punkt ersetzt.

Die einzelnen Angaben der Headeranzeige haben folgende Bedeutung :

DS :
Diese Ziffer gibt die Aufzeichnungsrate der Spur an. Die Zahlen 1,2,3 und 4 stehen dabei für 31250,33333,35714 und 38461 Bytes pro Sekunde. Diese Rate ist auf den äußeren Spuren größer als auf den Inneren, da diese länger sind (größerer Umfang).

TR/SE :
Nummer der ausgewählten Spur (Track) und des Sektors. Die Eingabe eines Wertes in der Zeile mit POS wählt einen Sektor aus. Dabei kann es sich entweder um die Nummer aus dem Sektorheader handeln oder die Angabe gibt an der wievielste Sektor nach dem Indexloch gelesen werden soll. Die Angabe in der darunter liegenden Zeile gibt die im Sektorheader gespeicherten Werte wieder.

HE: Zeigt Kennzeichen für Header an (normal 8).

BL: Zeigt Kennzeichen für Datenblock an (normal 7).

CKS: Gibt den Wert der Prüfsumme des Sektorheaders und des Datenblocks an.

ID: Hier werden die zwei ID-Zeichen, die im Sektorheader vermerkt sind aufgeführt.

POS: Bezeichnet die Stelle im Datenblock, an der der Cursor steht.

Die Handhabung der Byterate (DS) ist etwas kompliziert. Wird eine Spur und ein Sektor angegeben und dieser gelesen (CTRL r), dann wird die auf dieser Spur normalerweise verwendete Byterate gebraucht. Die DS (Density) Auswahl erreicht man durch dreimaliges Drücken der 'Home'-Taste. Wenn man den gleichen Sektor nochmal liest, geschieht dies mit der neuen Byterate. Man muß also beachten, daß jedesmal, wenn die Spur- oder Sektornummer geändert wird, automatisch die Byterate feststeht. Soll sofort mit einer anderen Byterate gearbeitet werden, muß man eben den DS-Parameter ändern - aber immer erst nach Spur- und Sektoreingabe.

Befehle des Sektormoduls :

CTRL r (read): Sektor in den Rechner einlesen.

CTRL w (write): Schreibt Sektor auf die Diskette zurück.

CTRL s (sum):
Berechnet Prüfsumme des Sektorheaders und des Datenblocks.

CTRL f (fill):
Füllt einen Sektor oder einen Teilsektor mit einem Wert.

CTRL h (hire):
Sucht einen Wert oder String in einem Sektor oder einem Diskbereich.

CTRL c (copy):
Kopiert einen Sektor oder einen Teilsektor in einen anderen Sektor oder einen anderen Sektorteil.

CTRL t (trace):
Liest die Verkettungsbytes jedes Sektors in den Rechner ein (Dauer ca. 15 Sekunden). Mit den Befehlen 'CTRL n' und 'CTRL p' lässt sich der nächste oder der vorangehende Sektor der Verkettungsliste aufrufen.

CTRL z (zero):
Sucht nächsten Sektor, dessen Inhalt nicht Null ist, d.h. den nächsten Sektor, der Informationen enthält.

CTRL # : Schaltet die Ausgabe auf dezimal.

CTRL $: Schaltet die Anzeige auf hexadezimal.

CTRL " :
Waehlt bei hexadezimalem Ausgabemodus die ASCII-Darstellung aus.

CTRL ' :
Wählt statt ASCII-Zeichen Bildschirmcode an.

BAMMODUL

Nun handelt es sich um die BAM, die Blockbelegungstabelle der Diskette. Hier das Bildsschirmbild :

```
-----------------------------------------
CTRL : r w t a l                      D:8
---------------[BAM]--------------------
BLOCKS FREE: 0496   DISK TYPE: 1541
-----------------------------------------
     012345678901234567890 FREE   SC
 01  .....................   21    0
 02  ..*...*.*...s........   18    1
 03  ***s********ss*******    0    3
                 .
                 .
```

Die Sektoren jeder Spur sind in jeweils einer Zeile aufgeführt. Ein Stern bedeutet dabei, daß der Sektor belegt ist. Der Punkt steht für einen unbelegten Sektor. Ist ein Sektor mit 's' markiert, dann signalsiert das, daß dieser Sektor frei ist, aber Daten enthält. Es handelt sich vermutlich um einen gelöschten Sektor. Die letzten beiden Spalten geben die Anzahl der freien Sektoren der Spur und die Zahl der gelöschten Sektoren an. Soll ein Sektor in der BAM belegt oder freigegeben werden, genügt es an die entsprechende Stelle einen '*' oder einen '.' zu setzen. Nun noch die übrigen Befehle :

CTRL r (read): Liest BAM in den Rechner ein.

CTRL w (write): Schreibt BAM auf Diskette.

CTRL t (track):
Belegt oder gibt gesamte Spur frei, auf der Cursor steht.

CTRL a (all): Belegt oder gibt gesamte Diskette frei.

CTRL s (scratched):
Analysiert alle Spuren nach gelöschten Sektoren.

MEMORY MODUL

Das Memory Modul erlaubt den direkten Eingriff ins Disk-CPU-System. So können Programme im DOS-Puffer geschrieben werden. Weiter können Sie die Zeropage oder die I/O-Adressen ansprechen. Damit haben Sie völlige Gewalt über die Floppy. Aus diesen Möglichkeiten entsteht natürlich auch ein gewisser Anspruch an das Wissen des Benützers. Daher wendet sich diese Funktionsgruppe hauptsächlich an die Maschinensprache-Programmierer.

```
----------------------------------------
CTRL : s l f h t g                  D:8
---------------[MEMORY]-----------------
LOC: $c100
----------------------------------------
  c100  78            sei           ...
  c101  a9  f7        lda #$f7      ...
  c103  2d  00  1c    and $1c00     ...
  c106  48            pha           ...
  c107  a5  7f        lda $7f       ...
  c109  f0  05        beq $c110     ...
  c10b  68            pla           ...
  c10c  09  00        ora #$00      ...
  c10e  d0  03        bne $c113     ...
  c110  68            pla           ...
  c111  09  08        ora #$08      ...
  c113  8d  00  1c    sta $1c00     ...
  c116  58            cli           ...
  c117  60            rts           ...
  c118  78            sei           ...
  c119  a9  08        lda #$08      ...
```

In das Feld LOC kann man eine beliebige Speicheradresse eintragen. Ab dieser Stelle wird dann der Speicher angezeigt. Auch hier wird wieder das Scroll-Prinzip angewandt. Stößt man mit dem Cursor an den oberen oder unteren Rand, dann wird einfach in die entsprechede Richtung weitergelistet. Die Darstellung des Speichers orientiert sich dabei an der 6502-Assembler-Schreibweise. Der Diskmonitor versucht den Speicher-

inhalt als Opcode zu interpretieren und gibt die Speicherstellen entsprechend aus. Darüberhinaus werden die Speicherinhalte noch eine hexadezimal und als ASCII-Zeichen angezeigt. Jede dieser Ausgabeversionen kann editiert werden. Danach werden die entsprechenden anderen Ausgaben automatisch korregiert.
BEFEHLE des Memory Moduls :

CTRL s (save):
Speichert einen beliebigen Disk-CPU-Speicherbereich als Autostart-Datei auf der Diskette ab. Dies vereinfacht das Erstellen von DOS-Programmen erheblich.

CTRL l (load):
Lädt Autostart-Datei in den DOS-Speicher ohne diese zu starten.

CTRL f (fill):
Füllt einen Speicherbereich mit einem bestimmten Wert.

CTRL h (hire):
Sucht in einem Speicherbereich einen Wert oder einen String.

CTRL t (transfer):
Kopiert den Inhalt eines Speicherbereichs in einen beliebigen anderen Bereich.

CTRL g (go):
Startet Maschinenspracheprogramm im DOS-Speicher.
CP/M TRACK MODUL

```
-------------------------------------
CTRL :                            D:8
--------------[TRACK]----------------
TRACK : 01  SIDE : 0
-------------------------------------
 0000 40 41 42 43 00 00 00 00  @abc....
 0008 00 00 00 00 31 39 38 35  ....1985
 0010 00 00 00 SS SS SS SS 08  ........
 0018 12 04 01 41 42 0F 0F 00  ...ab...
 0020 00 00 00 00 00 00 00 SS  ........
 0028 SS SS SS 07 01 02 03 04  ........
                 .
                 .
                 .
```

Das Datenbyte in der linken oberen Ecke wird mit 'Home' erreicht. Betätigt man zweimal die 'Home'-Taste, dann gelangt der Cursor auf der Track-Angabe.

Die Byte-Werte $F5, $F6 und $F7 erzeugen beim Schreiben der Spur Bytes mit speziellen Taktbytes. Enthalten die Sektoren einer Spur Daten, die diesen speziellen Byte-Werten entsprechen, werden an deren Stelle Markierungen für die 'Index Mark' usw. geschrieben. In diesem Fall entspricht die geschriebene Spur nicht mehr der eingelesenen. Das Schreiben von Spuren unter CP/M ist also immer mit Vorsicht durchzuführen.

Befehle des Trackmoduls»:

CTRL r (read): Liest die Spur in den Rechner ein.

CTRL w (read): Schreibt die Spur auf die Diskette zurück.

CTRL n (new):
Formatiert eine Spur neu. Die dazu nötigen Angaben, wie Sektorzahl, Sektorgröße usw. werden vom Diskmonitor erfragt.

CTRL f (fill):
Beschreibt die gesamten Spur oder einen Teil der Spur mit einem Wert. Damit können zum Beispiel sogenannte Killertracks gelöscht oder hergestellt werden (Kopierschutz).

CTRL s (sectors):
Sektoren einer Spur oder eines Spurbereichs mit einem Wert füllen.

CTRL c (copy):
Kopiert einen Teil einer Spur an eine andere Stelle auf der Spur.

CTRL l (location):
Damit kann man den Beginn der Spur festslegen. Der Cursor muß dabei auf dem Byte stehen, das nach dem Indexloch liegen soll.

CP/M SEKTOR MODUL

```
---------------------------------------
CTRL :                              D:8
---------------[SECTOR]----------------
TR: 24 SE: 10  POS: 000
---------------------------------------
 HE: 008 CKS: 021 TR: 24 SE: 10 ID: ab
 BL: 007 CKS: 122
---------------------------------------
 00  40 41 42 43 00 00 00 00  @abc....
 08  00 00 00 00 31 39 38 35  ....1985
 10  E5 E5 E5 E5 E5 E5 E5 E5  ........
                .
                .
                .
```

Befehle des CP/M Sektormoduls :

CTRL r (read): Sektor in den Rechner einlesen.

CTRL w (write): Schreibt Sektor auf die Diskette zurück.

CTRL f (fill):
Füllt einen Sektor oder einen Teilsektor mit einem Wert.

CTRL h (hire):
Sucht einen Wert oder String in einem Sektor oder einem Diskbereich.

CTRL c (copy):
Kopiert einen Sektor oder einen Teilsektor in einen anderen Sektor oder einen anderen Sektorteil.

CTRL z (zero):
Sucht naechsten Sektor, dessen Inhalt nicht $E5 ist, das bedeutet den nächsten Sektor, der Informationen enthält.

CTRL # : Schaltet die Ausgabe auf dezimal.

CTRL $: Schaltet die Anzeige auf hexadezimal.

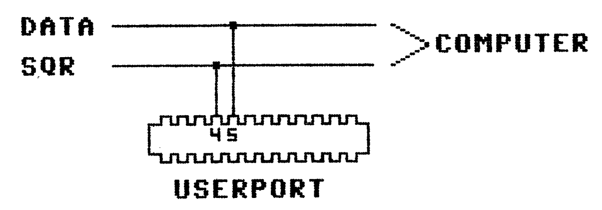
CTRL " : Wählt die ASCII-Darstellung im Hexmodus.

CTRL ' : Wählt statt ASCII-Zeichen Bildschirmcode an.

6.2 Der C-64 und die C-1570/71 - kein Problem

6.2.1 1570/71 Bus für den C-64 selbstgebastelt.

Sicherlich sind Sie als C-64 auf all die C-128 neidisch, weil diese die C-1570/71 mit dem schnellen Busmodus nützen können, der beim C-64 nicht vorhanden ist. Wenn es sich lediglich um ein Softwareproblem handeln würde, wäre es keine Schwierigkeit das System zu implementieren. Doch der neue Bus benötigt auch noch einige zusätzliche Hardware-Einrichtungen.



Der Clou ist allerdings, daß die Voraussetzungen für den schnellen Busmodus bereits im C-64 vorhanden sind. Das ist ein serielles Ein-/Ausgaberegister und die dazugehörige Taktleitung. Diese beiden Anschlüsse sind am User-Port des C-64 herausgeführt. Dabei sind diese Datenleitungen sowohl für CIA1 als auch CIA2 vorhanden.

Es ist am Besten, den seriellen Port des CIA1, das die Basisadresse $DC00 hat, zu verwenden. Somit ist die Ansteuerung des seriellen Registers genauso vorzunehmen wie beim C-128. Um nun die selbe Hardware, wie am C-128, zu erhalten, muß man nur die zwei Leitungen SRQ und Data mit den Pins 4 und 5 des User-Ports zu verbinden. Schon haben Sie einen seriellen Bus, der den C-128 Schnellmodus bedienen kann - eine Bastellösung für ein paar Mark.

6.2.2 Der Diskmonitor für den C-64

Der Diskmonitor aus Kapitel 6.1.1 kann auch in Zusammenhang mit dem C-64 benützt werden. Die einzige Einschränkung ist, daß die Spur-Schreiboperationen nicht funktionieren, da der C-64 dazu mit 2MHz Taktfrequenz betrieben werden müßte.

Alle anderen Funktionen des Diskmonitors stehen Ihnen auch mit einem C-64 und der C-1570/71 zur Verfügung, wenn Sie den im vorherigen Kapitel erläuterten Bus zusammenbasteln.

Die Diskette zum Buch enthält bereits eine fertig an den C-64 angepaßte Version des Diskmonitors.

6.2.3 Neues Turbo-Betriebssystem für den C-64

Das nachfolgende Programm ist eine Schnell-Lade-Routine, mit der Mit dem C-64 Programme genauso schnell geladen werden, wie mit dem C-128. Dabei ist dieses Programm in das Betriebssystem des C-64 eingebaut, wobei die Kassettenroutinen überschrieben werden und somit nicht mehr benutzt werden können. Das disassemlierte Listing zeigt die Funktion des Bus. Dabei muß Das Flag für den schnellen Busmodus in $033C gesetzt werden. Dies kann "von Hand" geschehen, wenn Sie immer eine C-1570/71 angeschlossen haben. Ist allerdings auch noch eine 1541-Floppy vorhanden, dann muß dieses Flag durch ein Programm gesetzt werden, das feststellt, ob das angesprochene Laufwerk im Schnellmodus laden kann. Einen derartigen Programmteil enthält beipielsweise das C-128 Kernal an der Stelle $E387.

Der Basic-Loader des Programms erstellt nach dem Start mit 'Run' eine spezielle Betriebssystem-Version im RAM des C-64, das die Fast-Load-Funktion enthält. Dafür ist der Kassettebetrieb abgeschalten.

Fast Load für C-64 (Kernal-Routine)

```
F72C  85 93     STA $93       Load-/Verify-Flag setzen
F72E  A9 00     LDA #$00      Statusbyte
F730  85 90     STA $90       löschen
F732  A5 BA     LDA $BA       Geräteadresse holen und auf
F734  C9 04     CMP #$04      IEC-Bus Adresse testen
F736  B0 03     BCS $F73B     ist IEC-Gerät angesprochen?
F738  4C A7 F4  JMP $F4A7     nein, zurück zum normalen Load
--------------------------------------------------------------------
Fast-Load vorbereiten
F73B  A4 B7     LDY $B7       Länge des Dateinamens holen
F73D  B1 BB     LDA ($BB),Y   Zeichen aus Dateinamen holen
F73F  C9 24     CMP #$24      mit '$' vergleichen
F741  F0 05     BEQ $F748     soll Directory geladen werden?
F743  88        DEY           nein, nächstes Zeichen anwählen
F744  10 F7     BPL $F73D     schon ganzer Name überprüft?
F746  30 03     BMI $F74B     ja, immer Sprung nach $F74B
F748  4C B8 F4  JMP $F4B8     zurück zur normalen Load-Routine
F74B  A4 B7     LDY $B7       Länge des Dateinamens holen
F74D  D0 03     BNE $F752     ist Name angegeben?
F74F  4C 10 F7  JMP $F710     nein, 'Missing Filename' ausgeben
F752  A5 BA     LDA $BA       Geräteadresse holen
F754  20 0C ED  JSR $ED0C     und LISTEN senden
F757  A9 6F     LDA #$6F      Sekundäradresse für Befehlskanal
F759  20 B9 ED  JSR $EDB9     Sekundäradresse senden
F75C  A5 90     LDA $90       Busstatus holen
F75E  10 06     BPL $F766     ist Floppy angeschlossen?
F760  20 2A F8  JSR $F82A     nein, Kanal wieder schließen
F763  4C 07 F7  JMP $F707     'Device Not Present' ausgeben
F766  A0 05     LDY #$05      Länge des Befehls für 1571-Modus
F768  B9 7E F8  LDA $F87E,Y   Zeichen aus Befehls-String holen
F76B  20 DD ED  JSR $EDDD     auf IEC-Bus ausgeben
F76E  88        DEY           nächstes Zeichen anwählen
F76F  D0 F7     BNE $F768     schon ganzer Befehl übertragen?
F771  20 54 F6  JSR $F654     ja, UNLISTEN senden
F774  A5 BA     LDA $BA       Geräteadresse der Floppy holen
F776  20 0C ED  JSR $ED0C     und LISTEN senden
F779  A9 6F     LDA #$6F      Sekundäradresse für Befehlskanal
F77B  20 B9 ED  JSR $EDB9     Sekundäradresse an Floppy senden
F77E  A0 03     LDY #$03      Länge des Fast-Load Befehls
```

```
F780  B9 7B F8   LDA $F87B,Y    Zeichen aus Befehls-String holen
F783  20 DD ED   JSR $EDDD      und auf IEC-Bus ausgeben
F786  88         DEY            Zeiger auf nächstes Zeichen richten
F787  D0 F7      BNE $F780      schon ganzer Befehl gesendet?
F789  B1 BB      LDA ($BB),Y    ja, Zeichen aus Dateiname holen
F78B  20 DD ED   JSR $EDDD      und auf IEC-Bus ausgeben
F78E  C8         INY            Zeiger auf nächstes Zeichen setzen
F78F  C4 B7      CPY $B7        auf Länge des Dateinamens prüfen
F791  D0 F6      BNE $F789      schon ganzer Name übertragen?
F793  20 54 F6   JSR $F654      ja, UNLISTEN senden
F796  2C 3C 03   BIT $033C      Flag für 'Fast-Bus Modus' testen
F799  10 06      BPL $F7A1      ist Fast-Bus möglich?
F79B  20 2A F8   JSR $F82A      nein, Kanal schließen
F79E  4C A7 F4   JMP $F4A7      Programm normal laden
-----------------------------------------------------------------
C-64 Fast-Load Routine
F7A1  78         SEI            Interruptroutine abschalten
F7A2  20 85 EE   JSR $EE85      Clock-Ausgang auf High setzen
F7A5  A9 88      LDA #$88       Flag für 'serielles Register voll'
F7A7  8D 0D DC   STA $DC0D      initialisieren
F7AA  AD 0E DC   LDA $DC0E      Kontrollregister holen
F7AD  29 80      AND #$80       Taktfrequenz für Echtzeituhr holen
F7AF  09 08      ORA #$08       Timer für Baudrate
F7B1  8D 0E DC   STA $DC0E      initialisieren
F7B4  2C 0D DC   BIT $DC0D      Flag für 'SP voll' zurücksetzen
F7B7  20 68 F8   JSR $F868      Clock-Ausgang umschalten
F7BA  20 71 F8   JSR $F871      Byte von 1571-Bus holen
F7BD  C9 02      CMP #$02       auf Code für 'File Not Found' prüfen
F7BF  D0 06      BNE $F7C7      identisch?
F7C1  20 2A F8   JSR $F82A      ja, Kanal schließen
F7C4  4C 04 F7   JMP $F704      'File Not Found' ausgeben
F7C7  48         PHA            Status-Code merken
F7C8  C9 1F      CMP #$1F       mit 'letzter Block' vergleichen
F7CA  D0 0B      BNE $F7D7      hat Datei nur einen Block?
F7CC  20 68 F8   JSR $F868      ja, Clock-Ausgang umschalten
F7CF  20 71 F8   JSR $F871      Byte vom 1571-Bus holen
F7D2  AA         TAX            Zahl der Datenbytes merken
F7D3  EA         NOP            2 Taktzyklen Verzögerung
F7D4  4C E4 F7   JMP $F7E4      Block laden
F7D7  C9 02      CMP #$02       Rückmeldung der Floppy prüfen
F7D9  90 09      BCC $F7E4      ist ein Fehler aufgetreten?
```

```
F7DB   68         PLA          ja, Stack wieder herstellen
F7DC   20 2A F8   JSR $F82A    Kanal schließen
F7DF   A0 00      LDY #$00     Nummer für 'I/O Error'
F7E1   4C 2B F1   JMP $F12B    Fehlermeldung ausgeben
F7E4   20 68 F8   JSR $F868    Clock-Ausgang umschalten
F7E7   20 71 F8   JSR $F871    Byte von 1571-Bus holen
F7EA   85 AE      STA $AE      Low-Byte der Startadresse setzen
F7EC   20 68 F8   JSR $F868    Clock-Ausgang umschalten
F7EF   20 71 F8   JSR $F871    Byte von 1571-Bus holen
F7F2   85 AF      STA $AF      High-Byte der Startadresse setzen
F7F4   A5 B9      LDA $B9      Sekundärangabe holen
F7F6   D0 08      BNE $F800    Programm relativ laden?
F7F8   A5 C3      LDA $C3      ja, Low-Byte des Basic-Speicher-
F7FA   85 AE      STA $AE      Beginns übernehmen
F7FC   A5 C4      LDA $C4      High-Byte des Speicherbeginns
F7FE   85 AF      STA $AF      setzen
F800   68         PLA          letzte Floppy-Rückmeldung holen
F801   C9 1F      CMP #$1F     auf 'letzter Sektor' prüfen
F803   F0 1D      BEQ $F822    war das der letzte Block?
F805   A2 FC      LDX #$FC     Zahl der Datenbytes setzen
F807   20 45 F8   JSR $F845    Block laden
F80A   20 71 F8   JSR $F871    Byte von 1571-Bus holen
F80D   C9 02      CMP #$02     Rückmeldung der Floppy prüfen
F80F   90 06      BCC $F817    ist Fehler aufgetreten?
F811   C9 1F      CMP #$1F     nein, auf 'letzter Block' prüfen
F813   F0 06      BEQ $F81B    folgt nun letzter Block?
F815   D0 C5      BNE $F7DC    nein, immer Sprung nach $F7DC
F817   A2 FE      LDX #$FE     Zahl der Datenbytes setzen
F819   D0 EC      BNE $F807    immer Sprung nach $F807
F81B   20 68 F8   JSR $F868    Clock-Ausgang umschalten
F81E   20 71 F8   JSR $F871    Byte von 1571-Bus holen
F821   AA         TAX          und als Anzahl der Datenbytes setzen
F822   20 45 F8   JSR $F845    Block laden
F825   A9 40      LDA #$40     Wert für 'EOF'
F827   20 1C FE   JSR $FE1C    Flag in Statusbyte setzen
F82A   20 85 EE   JSR $EE85    Clock-Ausgang auf High schalten
F82D   A9 81      LDA #$81     Interruptregister wieder
F82F   8D 0D DC   STA $DC0D    einrichten
F832   AD 0E DC   LDA $DC0E    Steuerregister holen
F835   29 80      AND #$80     Frequenz der Echtzeituhr holen
F837   09 11      ORA #$11     Timer für Interrupt
```

```
F839  8D 0E DC   STA $DC0E     setzen
F83C  58         CLI           Interrupt wieder einschalten
F83D  A5 B8      LDA $B8       Logische Kanalnummer
F83F  20 54 F6   JSR $F654     Kanal schließen
F842  4C A9 F5   JMP $F5A9     Load beenden
F845  20 68 F8   JSR $F868     Clock-Ausgang umschalten
F848  A0 00      LDY #$00      Zeiger löschen
F84A  A9 08      LDA #$08      Flag für 'SP voll' auswählen
F84C  2C 0D DC   BIT $DC0D     Flag testen
F84F  F0 FB      BEQ $F84C     ist serielles Register voll?
F851  AD 00 DD   LDA $DD00     ja, Bussteuerregister holen
F854  49 10      EOR #$10      und Clock-Ausgang
F856  8D 00 DD   STA $DD00     umschalten
F859  AD 0C DC   LDA $DC0C     Wert aus seriellem Register holen
F85C  91 AE      STA ($AE),Y   und in Speicher schreiben
F85E  E6 AE      INC $AE       Zeiger auf nächste Speicheradresse
F860  D0 02      BNE $F864     ist ein Übertrag entstanden?
F862  E6 AF      INC $AF       ja, High-Byte korregieren
F864  CA         DEX           Zahl der Datenbytes
F865  D0 E3      BNE $F84A     schon ganzer Block übertragen
F867  60         RTS           ja, zurück zur aufrufenden Routine
-----------------------------------------------------------------------
Signal am Clock-Ausgang umschalten
F868  AD 00 DD   LDA $DD00     Bussteuerregister holen
F86B  49 10      EOR #$10      und Clock-Ausgang
F86D  8D 00 DD   STA $DD00     umschalten
F870  60         RTS           zurück zur aufrufenden Routine
-----------------------------------------------------------------------
Byte vom 1571-Bus holen
F871  AD 0D DC   LDA $DC0D     Interruptregister holen
F874  29 08      AND #$08      und Flag für 'SP voll' testen
F876  F0 F9      BEQ $F871     ist Byte übertragen?
F878  AD 0C DC   LDA $DC0C     ja, Byte von Bus lesen
F87B  60         RTS           zurück zur aufrufenden Routine
```

Basic-Loader des Fast-Loads für C-64:

C-64 Fast-Load Loader

```
0800 00 0B 08 00 00 9E 32 30 36 31 00 00 00 78 A2 A0
0810 86 AF A0 00 84 AE B1 AE 91 AE C8 D0 F9 E8 86 AF
0820 E0 C0 D0 F2 A2 E0 86 AF B1 AE 91 AE C8 D0 F9 E8
0830 86 AF D0 F4 A9 08 8D 87 F3 A9 F7 8D B7 F4 8D F8
0840 F5 A9 7B 8D 5C 08 A9 08 8D 5D 08 A2 F7 8E 4D FD
0850 8E 60 08 A9 2C 8D 4C FD 8D 5F 08 B9 00 00 99 00
0860 00 C8 D0 F7 EE 5D 08 E8 8E 60 08 E0 F9 D0 EC A9
0870 05 85 01 8D D6 FD 20 15 FD 58 60 85 93 A9 00 85
0880 90 A5 BA C9 04 B0 03 4C A7 F4 A4 B7 B1 BB C9 24
0890 F0 05 88 10 F7 30 03 4C B8 F4 A4 B7 D0 03 4C 10
08A0 F7 A5 BA 20 0C ED A9 6F 20 B9 ED A5 90 10 06 20
08B0 2A F8 4C 07 F7 A0 05 B9 7E F8 20 DD ED 88 D0 F7
08C0 20 54 F6 A5 BA 20 0C ED A9 6F 20 B9 ED A0 03 B9
08D0 7B F8 20 DD ED 88 D0 F7 B1 BB 20 DD ED C8 C4 B7
08E0 D0 F6 20 54 F6 2C 3C 03 10 06 20 2A F8 4C A7 F4
08F0 78 20 85 EE A9 88 8D 0D DC AD 0E DC 29 80 09 08
0900 8D 0E DC 2C 0D DC 20 68 F8 20 71 F8 C9 02 D0 06
0910 20 2A F8 4C 04 F7 48 C9 1F D0 0B 20 68 F8 20 71
0920 F8 85 A5 4C E4 F7 C9 02 90 09 68 20 2A F8 A0 00
0930 4C 2B F1 20 68 F8 20 71 F8 85 AE 20 68 F8 20 71
0940 F8 85 AF A5 B9 D0 08 A5 C3 85 AE A5 C4 85 AF 68
0950 C9 1F F0 1D A2 FC 20 45 F8 20 71 F8 C9 02 90 06
0960 C9 1F F0 06 D0 C5 A2 FE D0 EC 20 68 F8 20 71 F8
0970 AA 20 45 F8 A9 40 20 1C FE 20 85 EE A9 81 8D 0D
0980 DC AD 0E DC 29 80 09 11 8D 0E DC 58 A5 B8 20 54
0990 F6 4C A9 F5 20 68 F8 A0 00 A9 08 2C 0D DC F0 FB
09A0 AD 00 DD 49 10 8D 00 DD AD 0C DC 91 AE E6 AE D0
09B0 02 E6 AF CA D0 E3 60 AD 00 DD 49 10 8D 00 DD 60
09C0 AD 0D DC 29 08 F0 F9 AD 0C DC 60 9F 30 55 31 4D
09D0 9E 30 55 00 00 00 00 00 00 00 00 00 00 00 00 00
```

KAPITEL 7
DAS DISK-OPERATING-SYSTEM (DOS)

7.1 Die DOS-Routinen

7.1.1 Das DOS - eine Einführung

Zuerst ein wenig Geschichte. Der Großvater des C-1570/71-DOS war das Betriebssytem der CBM 4040 Floppy. Diese Floppy war eine Doppelstation mit zwei Laufwerken. Dazu hatte die CBM 4040 eine Mikrocomputersteuerung mit zwei Prozessoren. Der eine Prozessor war für die Verwaltung der Daten und der andere für die Steuerung der Laufwerke zuständig. Diese Arbeitsteilung sollte das System schneller machen.

Als der VC-20, der erste Homecomputer von COMMODORE, auf den Markt kam, mußte dieser natürlich eine Floppy im Zubehörangebot haben. Das Entwicklung eines vollig neuen Systems wäre aufwendig gewesen und wozu soll man das Rad zweimal erfinden, wenn man bereits eine lauffähige Floppystation, die CBM 4040, zur Verfügung hat. Kurzum wurde eine Steuerelektronik mit einem Prozessor für ein Einzellaufwerk entwickelt. Die Software des 4040-Laufwerks, das DOS, wurde nur modifiziert. Die Verwaltungsroutinen sind die selben wie bei der CBM 4040. Nur den zweiten Prozessor für die Laufwerkssteuerung gab es beim neuen Einzellaufwerk nicht mehr. Also mußte der Prozessor des VC-1540-Laufwerks zusätzlich noch die Aufgaben des Steuerprozessors übernehmen. Das die VC-1540 dadurch nicht schneller wird braucht nicht erwähnt zu werden. Ein weiterer wichtiger Aspekt ist die Tatsache, daß die Verwaltung des Einzellaufwerks für ein Doppellaufwerk entworfen wurde.

Das DOS der VC-1541 ist mit dem der VC-1540 fast identisch. Hier wurden jediglich die Busroutinen geändert, da der C-64 eine etwas kleinere Taktfrequenz hat wie der VC-20 und deshalb die Bussteuerrung etwas langsamer abläuft.

Das C-1570/71 DOS besteht aus 16KByte-ROM der VC-1541 und weiteren 12 KByte neuen Betriebsystemteilen. Wieder hat die Faulheit gesiegt und COMMODORE hat einfach eine bereitsvorhandene DOS-Version etwas erweitert und zurechtgebogen. Das

dadurch die Leistungsfähigkeit der Floppy nicht gesteigert wird ist einsichtig.

Daüberhinaus ist mit dem C-1570/71-DOS V3.0 das Chaos perfekt. Dieses DOS enthält einen Verwaltungsteil, der für zwei Laufwerke und ein Multiprozessorsystem ausgelegt ist. Dazu kommt eine Laufwerksverwaltung, die für auch für Multiprozessor-Betrieb geeignet ist, allerdings nur ein Laufwerk ansteuern kann.

Dieser Teil des 1541-ROMs wurde komplett in das C-1570/71-ROM übernommen. Jediglich einige Routinen wurden etwas modifiiziert. Dabei wurden einfach neue Programmteile, die die zweite Diskettenseite verwalten usw. eingebaut. Dazu kommt eine neue Laufwerks-Controller-Routine, auch Jobschleife genannt. Zusätzlich zum VC-1541-ROM werden noch eine Reihe neuer Funktionen implementiert, die vorallem die Steuerung des Controllers WD 1770 übernehmen.

Summa summarum besteht das 1570/71 DOS aus einem Sammelsurium verschiedenster Programmteile, die irgendwie miteinander verknüpft sind. Das Traurige daran ist, daß aus diesem Grund die Floppy nicht besonders leistungsfähig ist, obwohl sie vielfältige technische Möglichkeiten bietet. Die langsame Übertragungs- und Laderaten sind nicht etwa die Schuld irgendwelcher Busverfahren, sondern alleinig das Produkt der langsamen DOS-Verwaltung.

7.1.2 Die wichtigsten DOS-Routinen

Das DOS besteht aus einer Unzahl verschiedener Routinen. Viele davon können Sie nicht benutzen, da sie nicht als Unterprogramme ausgelegt sind. Hier nun eine kleine Aufstellung einiger interessanter ROM-Routinen:

$8162 1571-Bus-Elektronik auf Eingang schalten.

$81CE 1571-Bus-Elektronik auf Ausgang schalten.

$85F9 Byte auf 1571 Bus ausgeben.
Das Byte muß in $46 gespeichert sein.

$864B Job ausführen.
$F9 muß die Nummer des Puffers enthalten, für den der Job ausgeführt werden soll. Die Jobtabelle $00-$05 enthält den Jobcode. Nach der Ausführung des Jobs enthalten X und A die Rückmeldung der Jobschleife.
Tritt bei der Ausführung ein Fehler auf, dann versucht die Routine noch einige mal den Job fehlerfrei auszuführen.

$8764 Laufwerksmotor einschalten.

$8770 Laufwerksmotor ausschalten.

$877C LED am Laufwerk einschalten.

$878B LED am Laufwerk ausschalten.

$883C Fehlerrückmeldung von WD 1770 holen.
A und X enthalten die Fehlernummer.

$884E Kommando in A an WD 1770 Befehlsregister übergeben.

$8861 Warten bis WD 1770 Befehl ausgeführt hat.

$89EF Kopf auf Spur0 zurückstellen.

$89FD Write Protect prüfen A= $10: kein Schreibschutz.
A= $00: Schreibschutz aktiv.

$9032 Auf 1541-Modus schalten.

$904E Auf 1571-Modus schalten.

$93F3 Kopf auf der aktuellen Diskettenseite aktivieren.
C=0: Seite 1 C=1: Seite2

$98D9 5 GCR-Bytes in 4 Binärbytes umrechnen.

$F6D0 4 Binärbytes in 5 GCR-Bytes umrechnen.

$FE00 Kopf auf Lesen umschalten.

$FE0E Spur löschen (mit $55 beschreiben)

7.1.3 Die Zeropage

Die Zeropage ist der Speicherbereich $0000 bis $00FF, der von bestimmten 6502-Assembler-Befehlen besonders schnell angesprochen werden kann. Deshalb werden in diesem Bereich die wichtigsten Parameter und Daten, die das DOS zum Arbeiten benötigt zwischengespeichert.

Eine besondere Bedeutung haben die Adressen $00 bis $11. Diese Speicherstellen werden dazu benützt an die Jobschleife, der Programmteil des DOS, der das Laufwerk steuert, Befehle und Parameter zu übergeben. Dabei ist für jeden Datenpuffer eine Speicherstelle reserviert, die das Kommando aufnimmt, welche Aktion für diesen Datenpuffer durchgeführt werden soll. Darüberhinaus meldet sich die Jobschleife mit einer Rückmeldung in diesen Speicherstellen, die darüber Aufschluß gibt, ob der Befehl fehlerfrei ausgeführt wurde.

Ein großer Teil der Zeropage wird für die Verwaltung der Dateien verwendet. Dabei wird dadurch, daß das DOS immer von zwei Laufwerken, also einer Doppelstation, ausgeht, sehr viel Platz für das zweite Laufwerk verschenkt, das gar nicht vorhanden ist.

Einige Zeropagestellen enthalten für den Betrieb der Floppy wichtige Konstanten. Dies sind zum Beipiel:

```
57  $39  Kennzeichen für Datenblock (8)
104 $68  Flag für Initialisierungsmethode.
106 $6A  Zahl der Leseversuche.
```

Diese Speicherstellen werden einmal beim Einschalten der Floppy gesetzt. Danach arbeitet das DOS immer mit den hier angegebenen Werten. Wenn Sie beispielsweise das Kennzeichen für den Datenblock verändern und auf diese Weise eine Diskette formatieren, kann die später mit dem normalen Kennzeichen nicht gelesen werden. Eine andere Möglichkeit ist, die Zahl der Leseversuche, sowie das Verhalten der Floppy bei Lesefehlern in Adresse $6A festzulegen.

Das ausführliche Zeropage-Listing enthält Kapitel 8. Sie müssen so nicht das ganze Buch nach dem Zeropage-Listing durchsuchen, sondern haben es immer am Ende des Buches griffbereit.

7.1.4 Das DOS V3.0 im Detail

Das DOS ist, wie schon in 7.1.1 erwähnt wurde, teilweise zum Handhaben von zwei Laufwerken oder für den Multiprozessorbetrieb geeignet. Da es nun aber auf einem System mit nur einem Laufwerk und nur einem Prozessor läuft, werden die Fähigkeiten des Systems bei weitem nicht ausgenutzt. Zum anderen ist für zwei Laufwerke eine aufwendigere Verwaltungsarbeit nötig, die bei dem Einzellaufwerk, der C-1570/71, nur zusätzliche Arbeitszeit benötigt, ohne einen Nutzen zu haben.

Besonders die Art und Weise der Pufferverwaltung, stammt noch aus den 4040-Doppellaufwerken. Dabei werden je nach Dateityp 1, 2 oder gar drei Puffer benötigt.

Da die Floppy bis zu 5 Dateien gleichzeitig verwalten kann, wird jede Datei einem internen Kanal zugewiesen. Diesem Kanal wiederum werden die benötigten Puffer zugeteilt. Darüberhinaus gibt es noch Verzeichnisse, die darüber Auskunft geben, welche Puffer gerade benötigt werden, welche Daten noch nicht bearbeitet sind usw.

Wie Sie sehen ist schon für den kleinsten Floppyzugriff eine enorme Verwaltungsarbeit nötig, die die Geschwindigkeit der C-1570/71 stark mindert.

Was beim DOS beachtet werden muß ist seine Entstehungsgeschichte. Dabei fällt besonders ins Gewicht, daß das DOS nicht in einem Zug entwickelt und durchdacht wurde, sondern immer nur aus den Versionen der Vorgänger-Floppies entstand und für die neue Floppy modifiziert wurde.

Mittlerweile ist diese DOS-Version drei- bis viermal verändert, ergänzt und erweitert worden. Dadurch steigt die Fehlerrate, die Zahl der unötigen Verwaltungsarbeiten und vorallem sinkt dadurch die Leistungsfähigkeit der Floppy.

7.1.5 Fehler im DOS

Selbstverständlich geht auch die Entwicklung eines Betriebssystems nicht fehlerfre vor sich. Und so hat der Fehlerteufel auch im 1570/71 DOS zugeschlagen. Das sind zum einen einige Funktionen und Befehle, die nicht in der gewünschten Art und Weise funktionieren, wie beispielweise der Block-Allocate-Befehl, die Replace-Funktion usw.

Darüberhinaus gibt es auch noch einige Stellen, an dehnen das ROM Befehle enthält, die im Zusammenhang fehlerhaft sind. Die umfangreichste Gruppe sind dabei die Assembler-Anweisungen, die keinen Sinn ergeben, überflüssig sind o.ä. Derartige Konstruktionen enthalten beispielsweise die Adresse:

$85DA $9396 $9690 $A605 $E853 $E9DA $EAA7 $F258 $FF13

Einige weitere DOS-Stellen sind allerdings fehlerhaft. Diese Fehler sind allerdings oft so geringfügig, daß sie sich nicht sofort in einem Floppy-Systemzusammenbruch äußern. Hier nun eine kleine Aufstellung einiger geheimnisvoller DOS-Stellen:

$8056: Hier werden aus $37 einige Flags ausgeblendet, die überhaupt keine Steuerfunktion haben. Das weißt darauf hin, daß

eigentlich ganz andere Flags ausgeblendet werden sollten, wobei es dann 'And $BE' heißen müßte.

$8124/$826F: Hier wird das Flag für die Echtzeituhr aktiviert, das im DOS überhaupt nicht verwendet wird. Da diese Aktion in Zusammenhang mit Busaktionen durchgeführt wird, liegt die Vermutung nahe, das eigentlich das danebenliegende Flag für das serielle Ein-/Ausgaberegister gemeint ist.

$BF57/BF75: Dies ist ein Jump an eine Stelle wo überhaupt kein Programm exisitiert.

$E69B: Bei dieser Routine wird der 'SED'-Befehl verwendet, ohne den Interrupt zu verhindern. So kommt es dann vor, das die Jobsteuerschleife aufgerufen wird, während BCD-Arithmetik akitiviert ist. Das dabei natürlich nicht die richtigen Steuerparameter errechnet werden, braucht nicht erwähnt zu werden.

7.2.1 Wie wurde kommentiert?

Das ROM-Listing in diesem Buch unterscheidet sich in einigen Punkten von vielen anderen ROM-Listings. Sicherlich werden Ihnen schon die merkwürdigen hochgestellten Zahlen nach der Speicheradresse aufgefallen sein, oder Sie haben sich schon gewundert, was die in eckigen Klammern angegebenen Zahlenkolonnen bedeuten. Bei diesen Dingen handelt es sich um eine sogenannte Cross-Referenz. Auf deutsch bedeutet das soviel wie Querverweise.

Die Angaben über jeder Routine des ROMs, die in eckigen Klammer stehen, nennen alle Stellen im ROM, bei dehnen an diese Routine aufgerufen wird. Wird dabei die Stelle aufgerufen an der die Werte in eckigen Klammern aufgeführt sind. Wird eine andere Stelle der Routine angesprungen, dann wird diese Einsprungadresse genannt, gefolgt von einem Doppelpunkt. Darauf ist die Adresse der Aufrufsstelle angegeben. Hierzu nun einige Beipiele:

[1234/5678]	Die ROM-Routine wird von 1234 und 5678 angeprungen.
[EEEA:1652]	Die Adresse EEEA der ROM-Routine wird von 1652 angesprungen.
[5527:78ED,5652]	Die Adresse 5527 wird von 78ED und 5652 angesprungen.

Neben diesen Querverweisen sind oft auch Kommentare angegeben, wenn die Stelle über einen Vektor oder eine Programm-Routine angesprungen wird. Genauso kommt auch hin und wieder der Kommentar 'Routine wird nicht verwendet' vor. Darüberhinaus werden bei einigen Routinen Vergleichsadressen angegeben, beispielsweise in der Form 'vgl. 1234'. Diese deuten darauf hin, daß diese selbe Routine oder eine Routine, die die gleiche Funktion hat an einer anderen Stelle im ROM vorkommt. Sie sollten diesen Hinweisen auf jeden Fall mit Interesse folgen. Sind zwei Routinen doppelt vorhanden, sind diese meistens nicht identisch kommentiert. Dadurch können Sie mit beiden Kommentarversionen arbeiten und haben so eine bessere, umfangreichere Erläuterung der ROM-Routine.

Eine andere Art von Querverweisen stellen die hochgestellten Zahlen, die hinter manchen Adressen angegeben sind, dar. Diese Ziffern geben an, daß an diese Stelle eingesprungen wird. Dabei handelt es sich meistens um relative Sprungbefehle. das bedeutet, die Stellen von dehnen diese gekennzeichnete Stelle aus angesprungen wird sind 128Bytes weiter vorne oder weiter hinten zu finden, was etwa einer Seite vor oder zurück entspricht. Die Zahl gibt dabei an, wieviele derartige Stellen existieren. Sollten Sie trotzdem den Einsprungpunkt nicht finden, kann es sein, daß die Daten dieser Adresse in der Kopfzeile in den eckigen Klammern angegeben ist. Prüfen Sie deshalb immer zuerst die Kopfzeile.

Was nützen diese Querverweise? Betrachten Sie dazu einmal die Adresse $93F3 im ROM-Listing. An dieser Stelle beginnt eine Routine mit einer Entscheidung. Nun stellt sich die Frage, wlchen Wert das Carryflag beim Einsprung hat. Die angegebenen Adressen erlauben die Stellen ausfindig zu machen, von dehnen aus die Routine $93F3 angesprochen wird. Somit kann auch festgestellt werden, welchen Wert das Carry-Flag hat.

Desweiteren zeigen Ihnen diese Querverweise, welche Routinenen sehr oft verwendet werden. Dabei werden Sie feststellen, daß die Hälfte aller DOS-Routinen nur einmal aufgerufen werden. Es handelt sich dabei im Grunde genommen nicht um Unterroutinen, sondern nur um Teilprogramme. Die hochgestellten Querverweiszahlen haben meistens den Wert 1. Diese Fälle sind aber nicht besonder interessant. Sie sollten Ihre Aufmerksamkeit auf alle Stellen lenken, die von mehr als einem Punkt aus angesprungen werden. Auf diese Weise können Sie den Programm-Ablauf einer Routine schneller erfassen.

Zum Schluß noch einige Worte zu den Kommentaren selbst. Es wurde versucht, alle Zeilen des ROM-Listings zu kommentieren. Bei einigen Stellen ist das problematisch, schließlich kann man keine spannenden Kommentare schreiben, wenn das Programm nicht spannender ist. An anderen Stellen wiederum reicht eine Zeile bbei weitem nicht aus, um die Routine zuu kommentieren. In diesen Fällen wurde öfters ein kleiner Abschnitt mit detaillierteren Erklärungen eingebaut.

Der Autor hat versucht, im gesamten ROM-Listing nur deutsche Kommentare zu benutzten. Dadurch soll das ROM-Listing verständlicher werden. Ein Fachjargon-Kauderwelsch hat keinen Nutzen, wenn es vom größten Teil der Leser nicht verstanden wird. Diejenigen, die dies verstehen würden, die Profies unter Ihnen, benötigen die ausführlichen Kommentare sowieso nicht, um das Betriebssystem zu verstehen.

7.2.2 C-1571 DOS LISTING
(ROM Version 03)

```
-----------------------------------------------------------------------
8000 92 25                    Prüfsumme des ROM [verwendet: 929D/92A4]
-----------------------------------------------------------------------
Copyright Vermerk
8002 53 2F 57 20 2D 20 44 41  S/W - DAVID G
800A 56 49 44 20 47 20 53 49  SIRACUSA
8012 52 41 43 55 53 41 0D 48  H/W - GREG
801A 2F 57 20 2D 20 47 52 45  BERLIN
8022 47 20 42 45 52 4C 49 4E  1985
802A 0D 31 39 38 35 0D
-----------------------------------------------------------------------
[CB63/806D:Vektor 80BE,80C0,80C6,80C8]
Routine für User-0-Befehl ('U0')
8030    AD 74 02  LDA $0274     Länge des Befehlsstrings holen
8033    C9 03     CMP #$03      und auf kleinste Befehlslänge prüfen
8035    90 2E     BCC $8065     ist der Befehl kleiner 3 Zeichen ?
8037    AD 02 02  LDA $0202     nein, Befehlsnummer holen
803A    85 3B     STA $3B       und merken
803C    29 1F     AND #$1F      Wert auf Bereich von 0-31 beschränken
803E    AA        TAX           und Steuerflags ausblenden
803F    0A        ASL A         Wert verdoppeln (da 2-Byte-Zeiger in
8040    A8        TAY           Tabelle) und als Zeiger setzen
8041    B9 8E 80  LDA $808E,Y   Low-Byte der angesprochenen Routine
8044    85 75     STA $75       holen und setzen
8046    B9 8F 80  LDA $808F,Y   High-Byte der Adresse
8049    85 76     STA $76       in Zeiger $75/$76 übernehmen
804B    E0 1E     CPX #$1E      auf 1541-Zusatzbefehle prüfen
804D    F0 07     BEQ $8056     sollen neue Befehle ausgeführt werden ?
804F    AD 0F 18  LDA $180F     nein, Flag für 1571/1541-Betrieb holen
8052    29 20     AND #$20      und überprüfen
8054    F0 0F     BEQ $8065     ist die Floppy im 1571-Modus ?
8056[1] A5 37     LDA $37       ja, [Fehler siehe 7.1.5]
8058    29 EB     AND #$EB      [unsinnige Bitflags werden]
805A    85 37     STA $37       [ausgeblendet]
805C    BD 6E 80  LDA $806E,X   Jobcode des äquivalenten Diskkontroller-
805F    8D 02 02  STA $0202     Befehls setzen
```

```
8062  6C 75 00    JMP ($0075)     Routine aufrufen
8065  A9 EA       LDA #$EA        Zeiger auf Tabelle
8067  85 6B       STA $6B         der 1541 User Befehle
8069  A9 FF       LDA #$FF        auf $FFEA
806B  85 6C       STA $6C         setzen
806D  60          RTS             zurück zur aufrufenden Routine
-----------------------------------------------------------------------
[805C]  Jobcodes zu den Befehlsroutinen
806E 80 81 90 91 B0 B1 F0 F1      Bit0 = Laufwerksnummer
8076 00 01 B0 01 00 01 00 01      Bit1-7 : $80 = Lesen / $90 = Schreiben
807E 80 81 90 91 B0 B1 F0 F1      $B0 = Sektor suchen  / $F0 = Formatieren
8086 00 01 B0 01 00 01 00 80      $00 = kein Job (Sonderfunktion)
-----------------------------------------------------------------------
[8041]  Adressen der Befehlsroutinen über 'User0'
        Befehlsnummer :
        Bit0   : Laufwerk (0/1)
        Bit1-3 : Funktion
        Bit4   : Diskettenseite (0/1)
808E 71 83       0 / 00        $8371  CP/M Sektor lesen
8090 7F 83       1 / 01        $837F  Fehlermeldung 'Drive Not Ready'
8092 EC 83       2 / 02        $83EC  CP/M Sektor schreiben
8094 F8 83       3 / 03        $83F8  Fehlermeldung 'Drive Not Ready'
8096 8B 84       4 / 04        $848B  CP/M Sektorheader lesen
8098 7F 83       5 / 05        $837F  Fehlermeldung 'Drive Not Ready'
809A B7 84       6 / 06        $84B7  CP/M Diskette formatieren
809C B7 84       7 / 07        $84B7  CP/M Diskette formatieren
809E F1 84       8 / 08        $84F1  CP/M Sektorversatz holen / setzen
80A0 F1 84       9 / 09        $84F1  CP/M Sektorversatz holen / setzen
80A2 17 85      10 / 0A        $8517  CP/M Sektorenfolge feststellen
80A4 7F 83      11 / 0B        $837F  Fehlermeldung 'Drive not ready'
80A6 6B 85      12 / 0C        $856B  Befehls-Statusbyte holen / setzen
80A8 7F 83      13 / 0D        $837F  Fehlermeldung 'Drive Not Ready'
80AA A5 85      14 / 0E        $85A5  'Syntax Error (31)' ausgeben
80AC A5 85      15 / 0F        $85A5  'Syntax Error (31)' ausgeben
80AE 71 83      16 / 10        $8371  CP/M Sektor lesen
80B0 7F 83      17 / 11        $837F  Fehlermeldung 'Drive Not Ready'
80B2 EC 83      18 / 12        $83EC  CP/M Sektor schreiben
80B4 F8 83      19 / 13        $83F8  Fehlermeldung 'Drive Not Ready'
80B6 8B 84      20 / 14        $848B  CP/M Sektorheader lesen
80B8 7F 83      21 / 15        $837F  Fehlermeldung 'Drive Not Ready'
80BA B7 84      22 / 16        $84B7  CP/M Diskette formatieren
```

```
80BC B7 84        23 / 17         $84B7  CP/M Diskette formatieren
80BE 6D 80        24 / 18         $806D  keine Funktion (rts)
80C0 6D 80        25 / 19         $806D  keine Funktion (rts)
80C2 17 85        26 / 1A         $8517  CP/M Sektorenfolge feststellen
80C4 7F 83        27 / 1B         $837F  Fehlermeldung 'Drive Not Ready'
80C6 6D 80        28 / 1C         $806D  keine Funktion (rts)
80C8 6D 80        29 / 1D         $806D  keine Funktion (rts)
80CA E5 8F        30 / 1E         $8FE5  1571-Zusatzkommandos ausführen
80CC 80 90        31 / 1F         $9080  Datei über 1571 Bus laden
-----------------------------------------------------------------------
[A7BA]
Kommando vom seriellen Bus annehmen (ATN aufgetreten)
80CE   78         SEI             Bus- und Kontrolleraufruf abschalten
80CF   A9 00      LDA #$00        Zeiger und Flags löschen :
80D1   85 7C      STA $7C         Flag für ATN vom Bus empfangen
80D3   85 79      STA $79         Flag für Listen
80D5   85 7A      STA $7A         Flag fur Talk
80D7   A2 45      LDX #$45        Stackpointer
80D9   9A         TXS             initialisieren
80DA   20 B2 81   JSR $81B2       1571 Bus auf Eingang schalten
80DD   A9 80      LDA #$80        Flag für letztes Zeichen (EOI = End
80DF   85 F8      STA $F8         of Information) setzen
80E1   85 7D      STA $7D         Flag für 'ATN beachten' löschen
80E3   20 B7 E9   JSR $E9B7       Clock-Ausgang auf Low
80E6   20 A5 E9   JSR $E9A5       Data-Ausgang auf High
80E9   AD 00 18   LDA $1800       Bussteuerregister holen und
80EC   09 10      ORA #$10        ATN-Ausgang auf High
80EE   8D 00 18   STA $1800       setzen
80F1   AD 00 18   LDA $1800       ATN-Eingang prüfen
80F4   10 64      BPL $815A       ist ATN gesetzt ?
80F6   29 04      AND #$04        ja, Clock-Eingang holen
80F8   D0 F7      BNE $80F1       ist Clock gesetzt ?
80FA   20 CA 82   JSR $82CA       nein, Kommandobyte vom Bus holen
80FD   C9 3F      CMP #$3F        mit Wert für 'Unlisten' vergleichen
80FF   D0 0C      BNE $810D       soll Listener seine Arbeit beenden ?
8101   A5 37      LDA $37         ja, Bussteuerflags holen
8103   29 BF      AND #$BF        und Flag für '1541-Bus-Modus'
8105   85 37      STA $37         setzen
8107   A9 00      LDA #$00        Flag für Listen
8109   85 79      STA $79         löschen
810B   F0 0E      BEQ $811B       immer Sprung nach $811B
```

```
810D[1] C9 5F      CMP #$5F     mit Wert für 'Untalk' vergleichen
810F    D0 0D      BNE $811E    soll Talker seine Arbeit beenden ?
8111    A5 37      LDA $37      ja, Bussteuerflags holen
8113    29 BF      AND #$BF     und Flag für '1541-Bus-Modus'
8115    85 37      STA $37      setzen
8117    A9 00      LDA #$00     Flag für Talk
8119    85 7A      STA $7A      löschen
811B[1] 4C 92 81   JMP $8192    warten bis ATN-Modus vorbei ist
811E[1] C5 78      CMP $78      mit Geräteadresse für Talk vergleichen
8120    D0 0A      BNE $812C    ist Talk adressiert ?
8122    A9 01      LDA #$01     ja, Flag für
8124    85 7A      STA $7A      'Talk empfangen' setzen
8126    A9 00      LDA #$00     Flag für Listen
8128    85 79      STA $79      löschen
812A    F0 29      BEQ $8155    immer Sprung nach $8155
812C[1] C5 77      CMP $77      mit Geräteadresse für Listen vergleichen
812E    D0 0A      BNE $813A    ist Listen adressiert ?
8130    A9 01      LDA #$01     ja, Flag für 'Listen empfangen'
8132    85 79      STA $79      setzen
8134    A9 00      LDA #$00     Flag für Talk
8136    85 7A      STA $7A      löschen
8138    F0 1B      BEQ $8155    immer Sprung nach $8155
813A[1] AA         TAX          ATN-Kommando merken
813B    29 60      AND #$60     Steuerbits für Talk und Listen isolieren
813D    C9 60      CMP #$60     und auf Wert für 'beide gesetzt' prüfen
813F    D0 4C      BNE $818D    wird Kanalnummer übermittelt ?
8141    8A         TXA          ja, Orginal-Sekundäradresse wieder holen
8142    85 84      STA $84      und setzen
8144    29 0F      AND #$0F     Nummer des angesprochenen Floppykanals
8146    85 83      STA $83      herstellen und setzen
8148    A5 84      LDA $84      Orginal-Sekundäradresse (ATN-Kommando)
814A    29 F0      AND #$F0     holen und Befehlbits isolieren
814C    C9 E0      CMP #$E0     ist Bit7 (Open/Close) auch gesetzt ?
814E    D0 42      BNE $8192    soll der Kanal geschlossen werden ?
8150    58         CLI          ja, Bus- u. Kontrolleraufruf einschalten
8151    20 C0 DA   JSR $DAC0    Kanal und evtl. Datei schließen
8154    78         SEI          Bus- und Kontrolleraufruf abschalten
8155[2] 2C 00 18   BIT $1800    ATN-Eingang testen
-----------------------------------------------------------------
```

```
[A9B3]  Bus nach ATN-Kommando steuern
8158     30 A0      BMI $80FA       ist immer noch ATN-gesetzt ?
815A[3]  A9 00      LDA #$00        nein, Flag für 'ATN aktiv'
815C     85 7D      STA $7D         löschen
815E     AD 00 18   LDA $1800       Bussteuerregister holen
8161     29 EF      AND #$EF        und ATN-Ausgang wieder
8163     8D 00 18   STA $1800       löschen
8166     A5 79      LDA $79         Flag für Listen holen
8168     F0 0D      BEQ $8177       ist der Bus im Listener Betrieb ?
816A     24 37      BIT $37         nein, Bussteuerflags testen
816C     50 03      BVC $8171       ist Bus im 1571 Modus ?
816E     20 99 81   JSR $8199       ja, DRF-Kode senden
8171[1]  20 42 83   JSR $8342       Daten vom Bus übernehmen
8174     4C 6B 83   JMP $836B       zur Befehls-Warteschleife
8177[1]  A5 7A      LDA $7A         Flag für Talk holen
8179     F0 0F      BEQ $818A       ist der Bus im Talker Betrieb
817B     20 9C E9   JSR $E99C       Data-Ausgang auf Low setzen
817E     20 AE E9   JSR $E9AE       Clock-Ausgang auf High setzen
8181     20 83 A4   JSR $A483       ca. 80 Taktzyklen Verzögerung
8184     20 EA 81   JSR $81EA       Daten nach Talk auf Bus geben
8187     20 83 A4   JSR $A483       ca. 80 Taktzyklen Verzögerung
818A[1]  4C 66 83   JMP $8366       zur Befehls-Warteschleife
818D[1]  A9 10      LDA #$10        ATN-Ausgang auf High
818F     8D 00 18   STA $1800       Data und Clock auf Low setzen
8192[3]  2C 00 18   BIT $1800       ATN-Eingang testen
8195     10 C3      BPL $815A       ist ATN gesetzt ?
8197     30 F9      BMI $8192       ja, warten bis ATN wieder gelöscht
--------------------------------------------------------------------------

[816E/81A1]
DRF (Device Request Fast) an Rechner senden (schnellen Busmodus anfordern)
8199     20 59 EA   JSR $EA59       auf ATN-Komandomodus prüfen
819C     20 C0 E9   JSR $E9C0       Busregister bei konstantem Wert lesen
819F     29 04      AND #$04        und Clock-Eingang isolieren
81A1     D0 F6      BNE $8199       ist Clock gesetzt ?
81A3     20 CE 81   JSR $81CE       nein, 1571 Bus auf Ausgang schalten
81A6     A9 00      LDA #$00        DRF-Signal
81A8     8D 0C 40   STA $400C       in serielles Ausgaberegister
81AB     A9 08      LDA #$08        Bitflag für serielles Register leer
81AD[1]  2C 0D 40   BIT $400D       Status der Ausgabe holen
81B0     F0 FB      BEQ $81AD       ist Datenbyte übertragen ?
--------------------------------------------------------------------------
```

```
[80DA/836B/846D/8591/8EAC/A61F/A7AD]
1571-Bus auf Eingang schalten
81B2  08        PHP         Prozessorstatus retten
81B3  78        SEI         Bus- und Kontrolleraufruf abschalten
81B4  AD 0E 40  LDA $400E   ja, Steuerregister holen
81B7  29 BF     AND #$BF    serielle Verbindung als Eingang
81B9  8D 0E 40  STA $400E   schalten
81BC  AD 0F 18  LDA $180F   Steuerbit für
81BF  29 FD     AND #$FD    1571-Busrichtung auf
81C1  8D 0F 18  STA $180F   Eingangsmodus setzen
81C4  A9 84     LDA #$84    [Fehler siehe 7.1.5]
81C6  8D 0D 40  STA $400D   [Echtzeituhr wird nicht verwendet]
81C9  2C 0D 40  BIT $400D   letzte Interruptflags zurücksetzen
81CC  28        PLP         Prozessorstatus wieder setzen
81CD  60        RTS         zurück zur aufrufenden Routine
--------------------------------------------------------------------
[81A3/8394/8461/84F6/8533/8582/8E93/8E9A/9080]
1571-Bus auf Ausgang schalten
81CE  08        PHP         Prozessorstatus retten
81CF  78        SEI         Bus- und Kontrolleraufruf abschalten
81D0  AD 0F 18  LDA $180F   Steuerbit für
81D3  09 02     ORA #$02    1571-Busrichtung auf
81D5  8D 0F 18  STA $180F   Ausgangsmodus setzen
81D8  AD 0E 40  LDA $400E   serielles Register
81DB  09 40     ORA #$40    als Ausgang
81DD  8D 0E 40  STA $400E   schalten
81E0  A9 08     LDA #$08    Interrupt bei 'Byte aus-/eingegeben'
81E2  8D 0D 40  STA $400D   verhindern
81E5  2C 0D 40  BIT $400D   Flags vom letzten Interrupt löschen
81E8  28        PLP         Prozessorstatus wieder setzen
81E9  60        RTS         zurück zur aufrufenden Routine
--------------------------------------------------------------------
[8184]  vgl. E909
Daten von Datei auf Bus ausgeben (1571 Modus)
81EA  78        SEI         Bus- und Kontrolleraufruf abschalten
81EB  20 EB D0  JSR $D0EB   Interne Kanalnummer feststellen
81EE  B0 06     BCS $81F6   wurde Kanal gefunden ?
81F0 1 A6 82    LDX $82     ja, Nummer des Kanals holen
81F2  B5 F2     LDA $F2,X   zugehörigen Busstatus feststellen
81F4  30 01     BMI $81F7   Kanal aktiv ?
81F6 1 60       RTS         nein, zurück zur aufrufenden Routine
```

```
81F7[1] 20 59 EA   JSR $EA59     auf ATN-Kommandomodus prüfen
81FA    20 C0 E9   JSR $E9C0     konstanten Wert vom Bussteuerregister
81FD    29 01      AND #$01      holen und Data-Eingang prüfen
81FF    08         PHP           Ergebnis merken
8200    20 B7 E9   JSR $E9B7     Clock-Ausgang auf High setzen
8203    28         PLP           Zustand der Data-Leitung wieder holen
8204    F0 12      BEQ $8218     ist Data gesetzt ?
8206[1] 20 59 EA   JSR $EA59     ja, auf ATN-Kommandomodus prüfen
8209    20 C0 E9   JSR $E9C0     Wert des Bussteuerregisters holen
820C    29 01      AND #$01      und Zustand des Data-Eingangs holen
820E    D0 F6      BNE $8206     ist Data immer noch gesetzt ?
8210    A6 82      LDX $82       nein, Nummer des aktuellen Kanals holen
8212    B5 F2      LDA $F2,X     und entsprechenden Busstatus holen
8214    29 08      AND #$08      EOI (End of Information) Flag testen
8216    D0 14      BNE $822C     ist letztes Byte übertragen worden ?
8218[2] 20 59 EA   JSR $EA59     nein, auf ATN-Kommandomodus prüfen
821B    20 C0 E9   JSR $E9C0     Wert des Bussteuerregisters holen und
821E    29 01      AND #$01      Zustand des Data-Eingangs feststellen
8220    D0 F6      BNE $8218     ist Data gesetzt ?
8222[1] 20 59 EA   JSR $EA59     nein, auf ATN-Kommandomodus prüfen
8225    20 C0 E9   JSR $E9C0     Wert vom Bussteuerregister holen
8228    29 01      AND #$01      und Zustand des Data-Eingangs testen
822A    F0 F6      BEQ $8222     ist Data jetzt gesetzt ?
822C[2] 20 AE E9   JSR $E9AE     ja, Clock-Ausgang auf High setzen
822F    20 59 EA   JSR $EA59     auf ATN-Kommandomodus prüfen
8232    20 C0 E9   JSR $E9C0     Wert vom Bussteuerregister holen
8235    29 01      AND #$01      und Data-Eingang isolieren
8237    D0 F3      BNE $822C     ist Data gesetzt ?
8239    24 37      BIT $37       nein, Flag für Busmodus testen
823B    50 39      BVC $8276     ist Bus im 1571 Modus ?
-----------------------------------------------------------------------------

Byte auf 1571 Bus ausgeben
823D    AD 0F 18   LDA $180F     ja, 1571-Bus Elektronik auf
8240    09 02      ORA #$02      Ausgangsmodus
8242    8D 0F 18   STA $180F     schalten (Bit1 =1)
8245    AD 0E 40   LDA $400E     serielles Ausgaberegister
8248    09 40      ORA #$40      als Ausgang
824A    8D 0E 40   STA $400E     ansteuern
824D    2C 0D 40   BIT $400D     Interruptregister zurücksetzen
8250    A6 82      LDX $82       Nummer des aktuellen Kanals
8252    BD 3E 02   LDA $023E,X   zu übertragendes Datenbyte des Kanals
```

```
8255  8D 0C 40   STA $400C     holen und in serielles Ausgaberegister
8258  AD 0D 40   LDA $400D     Zustand des Ausgaberegisters holen
825B  29 08      AND #$08      und prüfen ob Ausgaberegister leer ist
825D  F0 F9      BEQ $8258     ist Byte übertragen ?
825F  AD 0E 40   LDA $400E     ja, serielles Register
8262  29 BF      AND #$BF      als Eingangsregister
8264  8D 0E 40   STA $400E     schalten
8267  AD 0F 18   LDA $180F     Bus-Elektronik wieder
826A  29 FD      AND #$FD      in Eingangsmodus
826C  8D 0F 18   STA $180F     zurücksetzen (Bit1 =0)
826F  A9 84      LDA #$84      [Fehler siehe 7.1.54]
8271  8D 0D 40   STA $400D     [Echtzeituhr wird nicht verwendet]
8274  D0 3C      BNE $82B2     immer Sprung nach $82B2
------------------------------------------------------------------

Byte auf 1541 Bus ausgeben
8276  A9 08      LDA #$08      Zahl der Bits pro Byte
8278  85 98      STA $98       in Zähler festlegen
827A  20 C0 E9   JSR $E9C0     Bussteuerregister holen
827D  29 01      AND #$01      und Data-Eingang prüfen
827F  D0 43      BNE $82C4     ist Data gesetzt ?
8281  A6 82      LDX $82       nein, Nummer des aktuellen Kanals holen
8283  BD 3E 02   LDA $023E,X   und zugehöriges Datenbyte feststellen
8286  6A         ROR A         ein Bit davon übernehmen und
8287  9D 3E 02   STA $023E,X   restliches Byte wieder merken
828A  B0 05      BCS $8291     ist Bit auf 1 ?
828C  20 A5 E9   JSR $E9A5     nein, Data-Ausgang auf High setzen
828F  D0 03      BNE $8294     immer Sprung nach $8294
8291  20 9C E9   JSR $E99C     Data-Ausgang auf Low schalten
8294  20 7E A4   JSR $A47E     ca. 45 Taktzyklen Verzögerung
8297  A5 23      LDA $23       Flag für 1541/1540 Busverzögerung
8299  D0 E6      BNE $8281     ist Bus im 1541 Modus ?
829B  20 83 A4   JSR $A483     ja, ca. 80 Taktzyklen Verzögerung
829E  20 B7 E9   JSR $E9B7     Clock-Ausgang auf Low setzen
82A1  20 7E A4   JSR $A47E     ca. 45 Taktzyklen Verzögerung
82A4  A5 23      LDA $23       Flag für 1541/1540 Busverzögerung
82A6  D0 03      BNE $82AB     ist Bus im 1541 Modus ?
82A8  20 83 A4   JSR $A483     ja, ca. 80 Taktzyklen Verzögerung
82AB  20 FB FE   JSR $FEFB     Clock auf High und Data auf Low setzen
82AE  C6 98      DEC $98       Zähler für zu übertragende Bits
82B0  D0 C8      BNE $827A     ist Byte übertragen ?
82B2  20 59 EA   JSR $EA59     ja, auf ATN-Kommandomodus prüfen
```

```
82B5  20 C0 E9   JSR $E9C0     Bussteuerregister holen
82B8  29 01      AND #$01      und Data-Eingang übernehmen
82BA  F0 F6      BEQ $82B2     ist Data gesetzt ?
82BC  58         CLI           ja, Bus- & Kontrolleraufruf einschalten
82BD  20 AA D3   JSR $D3AA     nächstes Byte aus Datei lesen
82C0  78         SEI           Bus- und Kontrolleraufruf abschalten
82C1  4C F0 81   JMP $81F0     und wieder auf Ausgabe vorbereiten
82C4  4C 62 83   JMP $8362     zurück zur Befehls-Warteschleife
---------------------------------------------------------------------
[8358]
Kommandobyte vom 1571-Bus lesen
82C7  2C 0D 40   BIT $400D     Interrupregister zurücksetzen
82CA  A9 08      LDA #$08      Zahl der zu übertragenden Bits pro Byte
82CC  85 98      STA $98       festlegen
82CE  20 59 EA   JSR $EA59     auf ATN-Kommandomodus testen
82D1  20 C0 E9   JSR $E9C0     und Bussteuerregister einlesen
82D4  29 04      AND #$04      Clock-Eingang prüfen
82D6  D0 F6      BNE $82CE     ist Clock gesetzt ?
82D8  20 9C E9   JSR $E99C     nein, Data-Ausgang auf High setzen
82DB  A9 01      LDA #$01      Data-Eingang in
82DD  2C 00 18   BIT $1800     Bussteuerregister testen
82E0  D0 FB      BNE $82DD     ist Data noch gesetzt ?
82E2  8D 05 18   STA $1805     Timer 1 (Highbyte) setzen (1)
82E5  20 59 EA   JSR $EA59     auf ATN-Kommandomodus prüfen
82E8  AD 0D 18   LDA $180D     Interruptflag für
82EB  29 40      AND #$40      'Timer 1 ablgelaufen' prüfen
82ED  D0 09      BNE $82F8     sind schon 256 Taktzyklen vergangen ?
82EF  20 C0 E9   JSR $E9C0     nein, Wert vom Bussteuerregister holen
82F2  29 04      AND #$04      und Clock-Eingang überprüfen
82F4  F0 EF      BEQ $82E5     ist Clock gesetzt
82F6  D0 19      BNE $8311     ja, immer Sprung nach $8311
82F8  20 A5 E9   JSR $E9A5     Data-Ausgang auf High setzen
82FB  A2 18      LDX #$18      Warteschleife :
82FD  CA         DEX           etwa 0.1 ms
82FE  D0 FD      BNE $82FD     abwarten
8300  20 9C E9   JSR $E99C     Data-Ausgang auf Low setzen
8303  20 59 EA   JSR $EA59     auf ATN-Kommandomodus prüfen
8306  20 C0 E9   JSR $E9C0     Wert des Bussteuerregisters holen
8309  29 04      AND #$04      und Clock-Eingang isolieren
830B  F0 F6      BEQ $8303     ist Clock noch gesetzt ?
830D  A9 00      LDA #$00      ja, Flag für 'letztes Byte empfangen'
```

```
830F     85 F8      STA $F8       (EOI) setzen
8311[3]  AD 00 18   LDA $1800     Wert des Data-Eingangs feststellen,
8314     49 01      EOR #$01      invertieren und
8316     AA         TAX           merken
8317     AD 0D 40   LDA $400D     Flag für 'serielles Eingaberegister
831A     29 08      AND #$08      voll' holen
831C     F0 08      BEQ $8326     wurde ein Byte empfangen ?
831E     AD 0C 40   LDA $400C     ja, Byte aus Register lesen
8321     85 85      STA $85       und als aktuelles Datenbyte merken
8323     4C 3C 83   JMP $833C     und Ende
8326[1]  8A         TXA           invertierten Data-Wert wieder holen
8327     4A         LSR A         und im Carry merken
8328     29 02      AND #$02      Clock-Eingang prüfen
832A     D0 E5      BNE $8311     war Clock gleichzeitig gesetzt ?
832C     66 85      ROR $85       ja, Data-Bit in Datenbyte übernehmen
832E[1]  20 59 EA   JSR $EA59     auf ATN-Kommandomodus prüfen
8331     20 C0 E9   JSR $E9C0     Bussteuerregister holen
8334     29 04      AND #$04      Clock-Eingang testen
8336     F0 F6      BEQ $832E     ist Clock noch gesetzt ?
8338     C6 98      DEC $98       nein, Zähler für Zahl der Datenbits
833A     D0 D5      BNE $8311     ist gesamtes Byte empfangen ?
833C[1]  20 A5 E9   JSR $E9A5     ja, Data-Ausgang auf Low setzen
833F     A5 85      LDA $85       Datenbyte holen
8341     60         RTS           zurück zur aufrufenden Routine
----------------------------------------------------------------------

[8171/835F]  vgl. EA2E
Byte vom Bus übernehmen
8342     78         SEI           Bus- und Kontrolleraufruf abschalten
8343     20 07 D1   JSR $D107     Interne Kanalnummer feststellen
8346     B0 05      BCS $834D     wurde Kanal gefunden ?
8348     B5 F2      LDA $F2,X     ja, Status des entsprechenden Kanals
834A     6A         ROR A         Flag für Schreibbetrieb testen
834B     B0 0B      BCS $8358     ist Kanal zum Schreiben geöffnet ?
834D[1]  A5 84      LDA $84       nein, aktuelle Sekundäradresse holen
834F     29 F0      AND #$F0      und Befehlsbits holen und auf
8351     C9 F0      CMP #$F0      auf 'Kanal schließen' prüfen
8353     F0 03      BEQ $8358     soll Kanal beendet werden ?
8355     4C 66 83   JMP $8366     nein, zurück zur Befehls-Warteschleife
8358[2]  20 C7 82   JSR $82C7     Byte vom 1571 Bus holen
835B     58         CLI           Bus- und Kontrolleraufruf einschalten
835C     20 B7 CF   JSR $CFB7     Byte in Datei schreiben
```

```
835F  4C 42 83   JMP $8342       weiter
--------------------------------------------------------------------
[82C4]
Bus zurücksetzen; zurück zur Befehlswarteschleife
8362  A9 00      LDA #$00        Busstatusbyte
8364  85 37      STA $37         löschen
--------------------------------------------------------------------
[818A/8355]  Bus zurücksetzen; Modus beibehalten
8366  A9 00      LDA #$00        Data- und Clock-Ausgang auf Low
8368  8D 00 18   STA $1800       setzen
--------------------------------------------------------------------
[8174/E698/E8EA/EA53]
auf nächsten Befehl warten
836B  20 B2 81   JSR $81B2       1571 Bus auf Eingangsbetrieb schalten
836E  4C E7 EB   JMP $EBE7       auf nächsten Befehl vom Rechner warten
--------------------------------------------------------------------
[Einsprung über Vektor in 808E/80AE durch Routine $8030]
CP/M Sektor lesen; vorher Fehlerprüfung
8371  8D 4D 02   STA $024D       Jobcode der Routine
8374  85 5F      STA $5F         aus Tabelle $806E merken
8376  AD 0D 18   LDA $180D       CA2 Eingang testen (Elektronik zeigt an,
8379  4A         LSR A           daß 'Write Protect' unterbrochen wurde)
837A  90 18      BCC $8394       wurde Diskette gewechselt ?
837C  A2 0B      LDX #$0B        ja, Fehlernummer für 'ID Mismatch Error'
837E  2C         .byte $2C       nächste 2 Bytes überspringen (Bitbefehl)
--------------------------------------------------------------------
[Einsprung über Vektor : 8090/8098/80A4/80A8/80B0/80B8/80C4 durch $8030]
Fehler 'drive not ready' ausgeben
837F  A2 4F      LDX #$4F        Fehlernummer für 'drive not ready'
--------------------------------------------------------------------
[83C7/844A/84B4/8E42/8384:8DBC]
Befehls-Statusflags einblenden und mit Fehler ausgeben
8381  20 E9 85   JSR $85E9       Byte für Ausgabe vorbereiten
8384  20 81 85   JSR $8581       Meldung auf 1571 Bus ausgeben
--------------------------------------------------------------------
[84EE]  eventuellen Fehler ausgeben (ansonsten Rücksprung)
8387  E0 02      CPX #$02        Nummer mit Wert für 'Ok' vergleichen
8389  B0 01      BCS $838C       ist ein Fehler gesetzt ?
838B  60         RTS             nein, zurück zur aufrufenden Routine
--------------------------------------------------------------------
```

```
[8389/8484/8568/875C]
Fehlermeldung ausgeben (Nummer in X)
838C  8A          TXA           Fehlernummer holen und
838D  29 0F       AND #$0F      reine Fehlernummer herstellen
838F  A2 00       LDX #$00      Puffernummer 0 setzen
8391  4C 0A E6    JMP $E60A     Klartextmeldung bereitstellen
-----------------------------------------------------------------

[837A]
CP/M Sektor lesen
8394  20 CE 81    JSR $81CE     1571 Bus auf Ausgabe schalten
8397  24 5E       BIT $5E       Befehls-Statusbyte holen
8399  10 05       BPL $83A0     ist Flag für IBM-34 Diskette gesetzt ?
839B  A9 09       LDA #$09      ja, Routine ab $8D67 ausführen
839D  4C E6 86    JMP $86E6     (IBM-System-34  Sektor lesen)
83A0[1] 20 3D C6  JSR $C63D     Commodore-Diskette initialisieren
83A3[1] 58        CLI           Bus- und Kontrolleraufruf einschalten
83A4  A5 3B       LDA $3B       Befehlsnummer holen und Flag für
83A6  29 20       AND #$20      'Sektor nicht lesen' testen
83A8  D0 26       BNE $83D0     soll nur der Puffer übertragen werden ?
83AA  AD 03 02    LDA $0203     nein, viertes Zeichen aus Befehlsstring
83AD  85 06       STA $06       holen und als Spurnummer übernehmen
83AF  AD 04 02    LDA $0204     fünftes Zeichen holen und als Sektor-
83B2  85 07       STA $07       nummer des Jobs setzen
83B4  A2 00       LDX #$00      Puffer Nummer 0 anwählen
83B6  A5 5F       LDA $5F       aktuellen Jobcode holen
83B8  95 00       STA $00,X     und an Jobschleife übergeben
83BA  20 5E 86    JSR $865E     Job ausführen
83BD  78          SEI           Bus- und Kontrolleraufruf abschalten
83BE  20 E9 85    JSR $85E9     Rückmeldung für Ausgabe vorbereiten
83C1  24 3B       BIT $3B       Flag für 'Fehlertest' prüfen
83C3  70 04       BVS $83C9     soll die Rückmeldung beachtet werden ?
83C5  E0 02       CPX #$02      ja, Rückmeldung des Jobs auf 'Ok' prüfen
83C7  B0 B8       BCS $8381     ist der Job fehlerfrei verlaufen ?
83C9[1] 20 F9 85  JSR $85F9     ja, Rückmeldung auf 1571-Bus ausgeben
83CC  A5 3B       LDA $3B       Flag für 'Puffer ausgeben' testen
83CE  30 0D       BMI $83DD     Puffer an Rechner übertragen ?
83D0[1] A0 00     LDY #$00      ja, Pufferzeiger zuf Beginn des Puffers
83D2[1] B9 00 03  LDA $0300,Y   Byte aus Puffer holen
83D5  85 46       STA $46       und als auszugebendes Zeichen setzen
83D7  20 F9 85    JSR $85F9     Zeichen auf 1571-Bus ausgeben
83DA  C8          INY           Pufferzeiger auf nächstes Byte richten
```

```
83DB  D0 F5     BNE $83D2     schon der ganze Puffer übertragen ?
83DD  CE 05 02  DEC $0205     ja, Zahl der zu lesenden Sektoren
83E0  F0 06     BEQ $83E8     schon alle Sektoren ?
83E2  20 1E 86  JSR $861E     nein, Nummer des nächsten Sektors setzen
83E5  4C A3 83  JMP $83A3     nächsten Sektor lesen
83E8  58        CLI           Bus- und Kontrolleraufruf einschalten
83E9  4C AF 85  JMP $85AF     neue Spur holen und setzen
-----------------------------------------------------------------
[Einsprung über Vektor in 8092/80B2 durch Routine $8030]
CP/M Sektor schreiben; vorher Fehlerprüfung
83EC  8D 4D 02  STA $024D     Jobcode merken
83EF  AD 0D 18  LDA $180D     CA2 Eingang testen (Elektronik zeigt an,
83F2  4A        LSR A         daß 'Write Protect' unterbrochen wurde)
83F3  90 0D     BCC $8402     wurde Diskette gewechselt ?
83F5  A2 0B     LDX #$0B      ja, Fehlernummer für 'ID Mismatch Error'
83F7  2C        .byte $2C     nächste 2 Bytes überspringen (Bitbefehl)
-----------------------------------------------------------------
[Einsprung über Vektor in 8094 durch Routine $8030]
83F8  A2 4F     LDX #$4F      Fehlernummer für 'drive not ready'
83FA  86 46     STX $46       als auszugebendes Zeichen setzen
83FC  A5 3B     LDA $3B       Flag für
83FE  09 08     ORA #$08      'Fehler aufgetreten'
8400  85 3B     STA $3B       in Befehlsnummer übertragen
-----------------------------------------------------------------
[83F3]
CP/M Sektor schreiben
8402  24 5E     BIT $5E       Befehls-Statusbyte prüfen
8404  10 05     BPL $840B     Flag für IBM-34 Diskette gesetzt ?
8406  A9 0A     LDA #$0A      ja, Routine ab $8DF6 ausführen
8408  4C E6 86  JMP $86E6     (IBM-System-34 CP/M Sektor schreiben)
840B  20 3D C6  JSR $C63D     Commodore-Diskette initialisieren
840E  A5 3B     LDA $3B       Flag für 'Puffer von Rechner einlesen'
8410  30 29     BMI $843B     ist Flag in Befehlsbyte gesetzt ?
8412  78        SEI           ja, Bus- und Kontrolleraufruf abschalten
8413  A0 00     LDY #$00      Pufferzeiger auf Pufferbeginn setzen
8415  AD 00 18  LDA $1800     Bussteuerregister holen
8418  49 08     EOR #$08      Zustand des Clock-Ausgang umschalten
841A  2C 0D 40  BIT $400D     Interruptregister zurücksetzen
841D  8D 00 18  STA $1800     neuen Clock-Ausgangswert setzen
8420  AD 00 18  LDA $1800     ATN-Eingang testen
8423  10 03     BPL $8428     ist ATN gesetzt ?
```

```
8425     20 59 EA   JSR $EA59     ja, auf ATN-Kommandomodus prüfen
8428[1]  AD 0D 40   LDA $400D     Flag für 'Byte im seriellen Register
842B     29 08      AND #$08      empfangen' prüfen
842D     F0 F1      BEQ $8420     wurde ein Byte eingelesen ?
842F     AD 0C 40   LDA $400C     ja, Byte holen
8432     99 00 03   STA $0300,Y   und in Puffer 0 schreiben
8435     C8         INY           Pufferzeiger auf nächstes Byte richten
8436     D0 DD      BNE $8415     Puffer schon voll ?
8438     20 B7 E9   JSR $E9B7     ja, Clock-Ausgang auf Low setzen
843B[1]  58         CLI           Bus- und Kontrolleraufruf einschalten
843C     A5 3B      LDA $3B       Befehlsbyte holen und Flag für
843E     29 20      AND #$20      'Sektor nicht schreiben' prüfen
8440     D0 37      BNE $8479     gesetzt ?
8442     A5 3B      LDA $3B       nein, Flag für 'Fehler aufgetreten'
8444     29 08      AND #$08      in Befehlsnummer überprüfen
8446     F0 05      BEQ $844D     soll Fehler ausgegeben werden ?
8448     A6 46      LDX $46       ja, Nummer des Fehlers holen
844A     4C 81 83   JMP $8381     und auf 1571-Bus ausgeben
844D[1]  AD 03 02   LDA $0203     viertes Zeichen aus Befehlsstring holen
8450     85 06      STA $06       und als Spur für Jobschleife setzen
8452     AD 04 02   LDA $0204     fünftes Zeichen aus Befehlsstring holen
8455     85 07      STA $07       und als Sektor für Jobschleife setzen
8457     A2 00      LDX #$00      Puffer 0 anwählen
8459     A9 90      LDA #$90      Jobcode für 'Sektor schreiben'
845B     95 00      STA $00,X     an Jobschleife übergeben
845D     20 5E 86   JSR $865E     und ausführen
8460     78         SEI           Bus- und Kontrolleraufruf abschalten
8461     20 CE 81   JSR $81CE     1571-Bus auf Ausgang schalten
8464     20 E9 85   JSR $85E9     Rückmeldung für Ausgabe vorbereiten
8467     20 F9 85   JSR $85F9     Byte auf 1571-Bus ausgeben
846A     20 A0 86   JSR $86A0     auf Umspringen von Clock warten
846D     20 B2 81   JSR $81B2     1571-Bus auf Eingang schalten
8470     58         CLI           Bus- und Kontrolleraufruf einschalten
8471     24 3B      BIT $3B       Flag für 'Fehler testen' prüfen
8473     70 04      BVS $8479     soll Fehlerrückmeldung beachtet werden ?
8475     E0 02      CPX #$02      ja, Fehlernummer testen
8477     B0 0B      BCS $8484     ist Job fehlerfrei verlaufen ?
8479[2]  CE 05 02   DEC $0205     ja, Zähler für Sektoren
847C     F0 09      BEQ $8487     noch einen Sektor ?
847E     20 1E 86   JSR $861E     ja, neue Sektornummer errechnen
8481     4C 12 84   JMP $8412     Routine nochmal durchlaufen
```

```
8484  4C 8C 83   JMP $838C       zurück zur Befehlswarteschleife
8488  58         CLI             Bus- und Kontrolleraufruf einschalten
8488  4C AF 85   JMP $85AF       neue Spur setzen und Ende
-----------------------------------------------------------------------------
[Einsprung über Vektor 8096/80B6 der Routine $8030]
nächsten CP/M Sektorheader lesen (zuerst System-34, dann Commodore-Format)
848B  AD 02 02   LDA $0202       Jobcode holen
848E  29 01      AND #$01        und Laufwerksnummer daraus herstellen
8490  D0 20      BNE $84B2       ist Laufwerk 0 angesprochen ?
8492  A9 01      LDA #$01        ja, Flag für 'Write Protect wurde unter-
8494  8D 0D 18   STA $180D       brochen' (Diskwechsel) löschen
8497  A9 05      LDA #$05        Routine ab $8A09 ausführen
8499  20 E6 86   JSR $86E6       (IBM-System-34 Sektorheader lesen)
849C  AE B0 01   LDX $01B0       Rückmeldung holen und mit
849F  E0 02      CPX #$02        Wert für 'Ok' vergleichen
84A1  90 11      BCC $84B4       ist ein Fehler aufgetreten ?
84A3  A2 00      LDX #$00        ja, Befehls-Statusbyte
84A5  86 5E      STX $5E         löschen
84A7  A9 B0      LDA #$B0        Jobcode für 'Sektor suchen'
84A9  8D 4D 02   STA $024D       merken und an
84AC  95 00      STA $00,X       Jobschleife übergeben
84AE  20 5E 86   JSR $865E       Job ausführen
84B1  2C         .byte $2C       nächste 2 Bytes überspringen (Bitbefehl)
84B2  A2 4F      LDX #$4F        Fehlernummer für 'drive not ready'
84B4  4C 81 83   JMP $8381       Rückmeldung ausgeben und nächsten Befehl
-----------------------------------------------------------------------------
[Einsprung über Vektor 809A/809C/80BA/80BC durch Routine $8030]
CP/M Diskette formatieren
84B7  AD 02 02   LDA $0202       Jobcode holen und
84BA  29 01      AND #$01        angesprochenes Laufwerk feststellen
84BC  D0 2B      BNE $84E9       soll auf Laufwerk 0 formatiert werden ?
84BE  AD 03 02   LDA $0203       ja, Flag für Diskettentyp holen
84C1  10 05      BPL $84C8       ist Commodore-Format gewünscht ?
84C3  A9 08      LDA #$08        nein, Diskette in 'IBM System 34'
84C5  4C E6 86   JMP $86E6       formatieren (Routine $8C57)
84C8  A9 00      LDA #$00        Befehls-Statusbyte
84CA  85 5E      STA $5E         löschen
84CC  85 FF      STA $FF         Laufwerksstatus auf 'bereit' setzen
84CE  AD 04 02   LDA $0204       fünftes Zeichen aus Befehlsstrings holen
84D1  85 12      STA $12         und als erstes ID-Zeichen übernehmen
84D3  AD 05 02   LDA $0205       sechstes Zeichen aus Befehlsstring holen
```

```
84D6  85 13     STA $13       und als zweites Zeichen der ID speichern
84D8  20 07 D3  JSR $D307     alle Kanäle schließen
84DB  A9 01     LDA #$01      Spurnummer 1 als
84DD  85 80     STA $80       aktuellen Spur setzen
84DF  A9 FF     LDA #$FF      Diskette in 1571/1541-Format formatieren
84E1  8D 98 02  STA $0298     Rückmeldung holen
84E4  20 89 A9  JSR $A989     und für Ausgabe vorbereiten
84E7  AA        TAX           Meldung auf 1571-Bus ausgeben und Ende
84E8  2C        .byte $2C     nächste 2 Bytes überspringen (Bitbefehl)
84E9  A2 4F     LDX #$4F      Fehlernummer für 'drive not ready'
84EB  20 E9 85  JSR $85E9     Byte für Ausgabe vorbereiten
84EE  4C 87 83  JMP $8387     Fehlermeldung bereitstellen
-------------------------------------------------------------------------
[Einsprung über Vektor 809C/80A0 durch Routine $8030]
CP/M Sektorversatz holen / setzen
84F1  78        SEI           Bus- und Kontrolleraufruf abschalten
84F2  24 3B     BIT $3B       Befehlsnummer testen
84F4  10 0A     BPL $8500     soll Sektorversatz gelesen werden ?
84F6  20 CE 81  JSR $81CE     ja, 1571-Bus auf Ausgang schalten
84F9  A5 3C     LDA $3C       Sektorversatz holen und
84FB  85 46     STA $46       als auszugebendes Byte speichern
84FD  4C F9 85  JMP $85F9     Byte auf 1571-Bus ausgeben
8500  AE 74 02  LDX $0274     Länge des Befehlsstrings feststellen
8503  E0 04     CPX #$04      und prüfen ob genau drei Zeichen lang
8505  B0 0A     BCS $8511     sind genau 3 Zeichen im Puffer ?
8507  A2 0E     LDX #$0E      ja, Fehlernummer für 'Syntax Error'
8509  20 E9 85  JSR $85E9     Fehler für Ausgabe vorbereiten
850C  A9 31     LDA #$31      Fehlermeldung
850E  4C C8 C1  JMP $C1C8     '31 Syntax Error' bereitstellen
8511  AD 03 02  LDA $0203     Byte aus Befehlsstring holen (4.Zeichen)
8514  85 3C     STA $3C       und als neuen Sektorversatz übernehmen
8516  60        RTS           zurück zur aufrufenden Routine
-------------------------------------------------------------------------
[Einsprung über Vektor in 80A2/80C2 durch Routine $8030]
CP/M Sektorheader einlesen und Sektorenfolge feststellen
8517  20 8B 84  JSR $848B     nächsten Header lesen
851A  24 5E     BIT $5E       Befehls-Statusbyte auf IBM Flag testen
851C  10 48     BPL $8566     ist Flag für IBM-34 Diskette gesetzt ?
851E  A9 0D     LDA #$0D      ja, Routine $8F5F ausführen
8520  20 E6 86  JSR $86E6     (Tabelle der Sektorenfolge erstellen)
8523  AE B0 01  LDX $01B0     Rückmeldung holen und mit
```

```
8526  E0 02     CPX #$02      Wert für 'Ok' vergleichen
8528  B0 08     BCS $8532     ist ein Fehler aufgetreten ?
852A  20 61 89  JSR $8961     nein, kleinsten und größten Sektor holen
852D  20 86 89  JSR $8986     Sektorversatz errechnen
8530  8A        TXA           Sektorversatz holen
8531  48        PHA           und merken
8532  78        SEI           Bus- und Kontrolleraufruf abschalten
8533  20 CE 81  JSR $81CE     1571 Bus auf Ausgang schalten
8536  A5 5E     LDA $5E       Befehlsstatusbyte holen und als
8538  85 46     STA $46       auszugebendes Zeichen setzen
853A  20 F9 85  JSR $85F9     Byte auf 1571-Bus ausgeben
853D  AE B0 01  LDX $01B0     Fehlernummer nochmals holen
8540  E0 02     CPX #$02      und mit Wert für 'Ok' vergleichen
8542  B0 23     BCS $8567     ist ein Fehler aufgetreten ?
8544  A5 97     LDA $97       nein, Zahl der Sektoren der Spur als
8546  85 46     STA $46       auszugebendes Zeichen setzen
8548  20 F9 85  JSR $85F9     Byte auf 1571-Bus ausgeben
854B  A5 67     LDA $67       Nummer der gelesenen Spur als
854D  85 46     STA $46       auszugebendes Zeichen setzen
854F  20 F9 85  JSR $85F9     Byte auf 1571-Bus ausgeben
8552  A5 60     LDA $60       kleinste Sektornummer als
8554  85 46     STA $46       auszugebendes Zeichen setzen
8556  20 F9 85  JSR $85F9     Byte auf 1571-Bus ausgeben
8559  A5 61     LDA $61       größte Sektornummer als
855B  85 46     STA $46       auszugebendes Zeichen setzen
855D  20 F9 85  JSR $85F9     Byte auf 1571-Bus ausgeben
8560  68        PLA           Sektorversatz holen und  als
8561  85 46     STA $46       auszugebendes Zeichen setzen
8563  20 F9 85  JSR $85F9     Byte auf 1571-Bus ausgeben
8566  60        RTS           zurück zur aufrufenden Routine
8567  68        PLA           Stack wieder herstellen
8568  4C 8C 83  JMP $838C     Fehlermeldung ausgeben
-----------------------------------------------------------------
[Einsprung über Vektor in 80A6 durch Routine $8030]
Befehls-Statusbyte holen / setzen; Fehlernummer holen
856B  24 3B     BIT $3B       Befehlsnummer testen
856D  10 27     BPL $8596     Befehls-Statusbyte holen ?
856F  24 3B     BIT $3B       Flags in Befehlsnummer testen
8571  50 0E     BVC $8581     soll Diskettenwechsel geprüft werden ?
8573  AD 0D 18  LDA $180D     ja, Signal von Elektronik für 'Write-
8576  4A        LSR A         Protect wurde unterbrochen' testen
```

```
8577  90 08     BCC $8581      wurde die Diskette gewechselt ?
8579  A5 5E     LDA $5E        Befehls-Statusbyte holen
857B  29 F0     AND #$F0       und Flags herstellen
857D  09 0B     ORA #$0B       Fehlernummer für 'ID Mismatch' setzen
857F  85 5E     STA $5E        in Statusbyte übernehmen
-------------------------------------------------------------------------
[8384/8571/8577]
Befehls-Statusbyte ausgeben
8581  78        SEI            Bus- und Kontrolleraufruf abschalten
8582  20 CE 81  JSR $81CE      1571 Bus auf Ausgang schalten
8585  A5 5E     LDA $5E        Statusbyte als auszugebendes
8587  85 46     STA $46        Zeichen setzen
8589  20 F9 85  JSR $85F9      Byte auf 1571 Bus ausgeben
858C  A9 00     LDA #$00       Flag für Fehler (Blinkzähler)
858E  8D 6C 02  STA $026C      löschen
8591  20 B2 81  JSR $81B2      1571 Bus auf Eingang schalten
8594  58        CLI            Bus- und Kontrolleraufruf einschalten
8595  60        RTS            zurück zur aufrufenden Routine
-------------------------------------------------------------------------
Befehls-Statusbyte setzen
8596  AD 03 02  LDA $0203      4. Zeichen aus Befehlsstring holen
8599  85 5E     STA $5E        und als Befehls-Status übernehmen
859B  24 3B     BIT $3B        Flags aus Befehlsnummer testen
859D  50 05     BVC $85A4      soll Diskwechsel beachtet werden ?
859F  A9 01     LDA #$01       ja, Flag für Diskwechsel (Write Protect
85A1  8D 0D 18  STA $180D      unterbrochen) initialisieren
85A4  60        RTS            zurück zur aufrufenden Routine
-------------------------------------------------------------------------
[Einsprung über Vektor in 80AA/80AC durch Routine $8030]
'Syntax Error' ausgeben
85A5  A2 0E     LDX #$0E       Fehlernummer setzen
85A7  20 E9 85  JSR $85E9      Byte für Ausgabe vorbereiten
85AA  A9 31     LDA #$31       Fehlermeldung
85AC  4C C8 C1  JMP $C1C8      '31 Syntax Error' ausgeben
-------------------------------------------------------------------------
[83E9/8488]
neue Spur ansteuern
85AF  AD 74 02  LDA $0274      Länge des Befehlsstring holen
85B2  C9 07     CMP #$07       und auf 7 Zeichen prüfen
85B4  90 32     BCC $85E8      hat Befehlsstring mind. 7 Zeichen ?
85B6  A5 06     LDA $06        ja, Nummer der letzten Spur holen
```

```
85B8  A8          TAY          und merken
85B9  E9 01       SBC #$01     daraus aktuelle Position des Kopfes
85BB  0A          ASL A        in Halbschritten errechnen
85BC  85 64       STA $64      und setzen
85BE  C0 24       CPY #$24     war letzte Spur auf der 2. Seite
85C0  08          PHP          Ergebniss der Prüfung merken
85C1  AC 06 02    LDY $0206    7. Zeichen aus Befehlsstring holen
85C4  84 22       STY $22      und als aktuelle Spur setzen
85C6  88          DEY          daraus Zielspur -1
85C7  84 67       STY $67      errechnen und setzen
85C9  C0 23       CPY #$23     ist neue Spur auf Seite 2 der Diskette ?
85CB  6A          ROR A        Ergebniss in Bit7 schieben
85CC  28          PLP          vorheriges Ergebniss wieder holen
85CD  29 80       AND #$80     und letztes Ergebniss bereitstellen
85CF  90 0B       BCC $85DC    letzte Spur auf Seite 2 (Bit =1) ?
85D1  30 12       BMI $85E5    ja, neue Spur auf Seite 2 (Bit =1) ?
85D3  18          CLC          nein, neue Spur
85D4  A5 67       LDA $67      auf Seite 2
85D6  69 23       ADC #$23     umrechnen und
85D8  85 67       STA $67      setzen
85DA  30 09       BMI $85E5    immer Sprung
85DC[1] 10 07     BPL $85E5    nach $85E5
85DE  38          SEC          neue Spurnummer
85DF  A5 67       LDA $67      auf Seite 1
85E1  E9 23       SBC #$23     umrechnen und
85E3  85 67       STA $67      merken
85E5[3] 4C BA 87  JMP $87BA    Spur ansteuern
85E8[1] 60        RTS          zurück zur aufrufenden Routine
--------------------------------------------------------------------
[8381/83BE/8464/84EB/8509/85A7/8D64/8DB1/8EA3]
Fehlerbyte für Ausgabe vorbereiten
85E9  86 46       STX $46      Fehlernummer merken
85EB  A5 5E       LDA $5E      Befehls-Statusbyte holen
85ED  29 F0       AND #$F0     und Flags isolieren
85EF  05 46       ORA $46      Fehlernummer übernehmen und Wert
85F1  85 5E       STA $5E      als neuen Status sowie
85F3  85 46       STA $46      als auszugebendes Zeichen setzen
85F5  60          RTS          zurück zur aufrufenden Routine
--------------------------------------------------------------------
[8603]  Byte auf 1571 Bus ausgeben
85F6  20 59 EA    JSR $EA59    auf ATN-Kommandomodus prüfen
```

```
---------------------------------------------------------------------
[83C9/83D7/8467/84FD/853A/8548/854F/8556/855D/8563/8589/85FF/8609]
[8DBF/8DD4/8EA6]
Byte auf 1571 Bus ausgeben
85F9    AD 00 18   LDA $1800      Bussteuerregister holen und warten
85FC    CD 00 18   CMP $1800      bis Leitungszustand konstant bleibt
85FF    D0 F8      BNE $85F9      liegt konstanter Wert an ?
8601    29 FF      AND #$FF       ja, Prozessorflags (N/Z) neu setzen
8603    30 F1      BMI $85F6      ist ATN gesetzt ?
8605    45 37      EOR $37        nein, Busstatus holen und
8607    29 04      AND #$04       Flag für Clock testen
8609    F0 EE      BEQ $85F9      ist Clock gesetzt ?
860B    A5 46      LDA $46        ja, auszugebendes Zeichen holen und
860D    8D 0C 40   STA $400C      in serielles Ausgaberegister übertragen
8610    A5 37      LDA $37        Flag für Clock-holen
8612    49 04      EOR #$04       und invertieren
8614    85 37      STA $37        Flag wieder speichern
8616    A9 08      LDA #$08       Bitflag für 'Register ausgegeben'
8618[1] 2C 0D 40   BIT $400D      überprüfen
861B    F0 FB      BEQ $8618      ist Byte vollständig ausgegeben ?
861D    60         RTS            ja, zurück zur aufrufenden Routine
---------------------------------------------------------------------
[83E2/847E]
Nummer des nächsten IBM-34-Sektors errechnen
861E    AD 03 02   LDA $0203      Spurnummer aus Befehlsstring holen
8621    C9 24      CMP #$24       und mit max. Spur +1 vergleichen
8623    90 02      BCC $8627      ist Spur auf Seite 2 ?
8625    E9 23      SBC #$23       ja, Spur auf Seite 1
8627[1] AA         TAX            umrechnen und setzen
8628    BD 2B 94   LDA $942B,X    Zahl der Sektoren der Spur feststellen
862B    AA         TAX            und merken
862C    CA         DEX            daraus max. Sektornummer herstellen
862D    86 46      STX $46        und merken
862F    18         CLC            neue Sektornummer setzen :
8630    AD 04 02   LDA $0204      Sektornummer aus Befehlsstring
8633    65 3C      ADC $3C        Sektorversatz dazurechnen
8635    C5 46      CMP $46        mit max. Nummer vergleichen
8637    90 0A      BCC $8643      ist der erlaubte Bereich überschritten ?
8639    E5 46      SBC $46        ja, Nummer auf gültigen Bereich setzen
863B    F0 04      BEQ $8641      letzten Sektor angewählt ?
863D    38         SEC            ja, dann Sektornummer errechnen
```

```
863E  E9 01     SBC #$01     (da Sektor 0 auch existiert )
8640  2C        .byte $2C    nächste 2 Bytes überspringen (Bitbefehl)
8641[1] A5 46   LDA $46      ersten errechneten Wert holen und
8643[1] 8D 04 02 STA $0204   aktuelle Sektornummer merken
8646  A9 88     LDA #$88     'Sektor auf aktueller Spur lesen'
8648  85 5F     STA $5F      als aktuellen Jobcode übergeben
864A  60        RTS          zurück zur aufrufenden Routine
------------------------------------------------------------------
[910E]
Job ausführen
864B  A6 F9     LDX $F9      Nummer des aktuellen Puffers
864D  08        PHP          Prozessorstatus retten
864E  58        CLI          Bus- und Kontrolleraufruf einschalten
864F  20 B6 9F  JSR $9FB6    Jobschleife starten und Job ausführen
8652  C9 02     CMP #$02     Rückmeldung mit 'Ok' vergleichen
8654  90 05     BCC $865B    ist Job fehlerfrei verlaufen ?
8656  20 83 86  JSR $8683    nein, weitere Versuche durchführen
8659  B5 00     LDA $00,X    Rückmeldung holen
865B[1] AA      TAX          und merken
865C  28        PLP          Prozessorstatus wieder herstellen
865D  60        RTS          zurück zur aufrufenden Routine
------------------------------------------------------------------
[83BA/845D/84AE]
Jobschleife starten und Job für Puffer 0 ausführen
865E  A2 00     LDX #$00     Puffernummer festlegen
8660  08        PHP          Prozessorstatus merken
8661  78        SEI          Bus- und Kontrolleraufruf abschalten
8662  AD 00 1C  LDA $1C00    Laufwerkssteuerregister holen
8665  09 08     ORA #$08     und Bit für LED setzen
8667  8D 00 1C  STA $1C00    LED anschalten
866A  58        CLI          Bus- und Kontrolleraufruf einschalten
866B  20 B6 9F  JSR $9FB6    Jobschleife starten und Job ausführen
866E  C9 02     CMP #$02     Rückmeldung auf 'Ok' prüfen
8670  90 03     BCC $8675    ist Job fehlerfrei verlaufen ?
8672  20 83 86  JSR $8683    nein, erneute Versuche ausführen
8675[1] 78      SEI          Bus und Kontrolleraufruf abschalten
8676  AD 00 1C  LDA $1C00    Steuerregister holen
8679  29 F7     AND #$F7     und LED-Bit löschen
867B  8D 00 1C  STA $1C00    LED ausschalten
867E  B5 00     LDA $00,X    Rückmeldung des letzten Versuchs
8680  AA        TAX          holen und merken
```

```
8681  28        PLP             Prozessorstatus wieder herstellen
8682  60        RTS             zurück zur aufrufenden Routine
-----------------------------------------------------------------
[8656/8672]
weitere Versuche den Job auszuführen
8683  A9 FF     LDA #$FF        Flag für
8685  8D 98 02  STA $0298       'Fehler bei Jobausführung' setzen
8688  86 F9     STX $F9         Puffernummer des Jobs merken
868A  AD 02 02  LDA $0202       Jobcode holen
868D  85 5F     STA $5F         und als aktuellen Jobcode
868F  8D 4D 02  STA $024D       merken
8692  9D 5B 02  STA $025B,X     Jobcode dem Puffer zuordnen
8695  85 00     STA $00         und als Job für Puffer 0
8697  20 B6 9F  JSR $9FB6       an Jobschleife übergeben
869A  4C 99 D5  JMP $D599       Jobausführung kontrollieren
-----------------------------------------------------------------
869D1 20 59 EA  JSR $EA59       auf ATN-Kommandomodus testen
-----------------------------------------------------------------
[846A/86A6/86B0/8EA9]
Auf Umspringen des Clock-Eingangs warten
86A0  AD 00 18  LDA $1800       Bussteuerregister holen
86A3  CD 00 18  CMP $1800       und mit Wert vom 2. Lesen vergleichen
86A6  D0 F8     BNE $86A0       ist Zustand des Registers konstant ?
86A8  29 FF     AND #$FF        Prozessorflags (N/Z) setzen
86AA  30 F1     BMI $869D       ist ATN gesetzt ?
86AC  45 37     EOR $37         nein, Clock-Wert mit Wert vom letzten
86AE  29 04     AND #$04        Aufruf der Routine vergleichen
86B0  F0 EE     BEQ $86A0       hat er sich geändert ?
86B2  A5 37     LDA $37         ja, dann Flag holen
86B4  49 04     EOR #$04        und auch umdrehen
86B6  85 37     STA $37         Flag wieder merken
86B8  60        RTS             zurück zur aufrufenden Routine
-----------------------------------------------------------------
```

[Tabelle wird verwendet in 86E9]
Steuerbytes für Funktionen vor Aufruf der IBM-34 Routinen

Funktion der einzelnen Bits (Bit=1 heißt 'Funktion aktivieren') :
bit0 Fehler bei Jobausführung nicht ausgeben
bit1 nächsten Header lesen und Kopf nach gelesener Spur setzen
bit2 Warten bis Motor läuft und Kopf auf Position ist (Spur und Seite)
bit3 Kopf auf neue Spur ($67) positionieren

bit4 Laufwerksmotor anschalten
bit5 Write Protect prüfen
bit6 Sektornummer aus Befehlsstring übernehmen
bit7 Wert für neue Spur setzen ($67)

86B9	00	%00000000	keine Zusatzfunktionen
86BA	15	%00010101	Motor an/warten/keine Fehlermeldung
86BB	00	%00000000	keine Zusatzfunktionen
86BC	00	%00000000	keine Zusatzfunktionen
86BD	00	%00000000	keine Zusatzfunktionen
86BE	15	%00010101	Motor an/warten/keine Fehlermeldung
86BF	00	%00000000	keine Zusatzfunktionen
86C0	BC	%10111100	Spur/WP prüfen/Motor an/Kopf/warten
86C1	34	%00110100	WP prüfen/Motor an/warten
86C2	DE	%11011110	Spur/Sektor/Motor an/Kopf/Header
86C3	FE	%11111110	Spur/Sektor/WP prüfen/Motor an/Kopf/Header
86C4	DC	%11011100	Spur/Sektor/Motor an/Kopf/warten
86C5	15	%00010101	Motor an/warten/keine Fehlermeldung
86C6	15	%00010101	Motor an/warten/keine Fehlermeldung
86C7	00	%00000000	keine Zusatzfunktionen

[Tabelle wird verwendet in 873F]
Adressen der Routinen für Diskettem im IBM-System-34 Format

86C8	EC 89	0 / 00	$89EC	Reset ausführen ($EAA0)
86CA	EF 89	1 / 01	$89EF	Kopf auf Spur 0 zurücksetzen
86CC	FD 89	2 / 02	$89FD	'Write Protect' prüfen
86CE	03 8A	3 / 03	$8A03	Spurparameter übernehmen
86D0	08 8A	4 / 04	$8A08	keine Funktion (rts)
86D2	09 8A	5 / 05	$8A09	nächsten IBM-34 Header lesen
86D4	BA 87	6 / 06	$87BA	keine Funktion (rts)
86D6	86 8A	7 / 07	$8A86	Spur 'IBM System 34' formatieren
86D8	57 8C	8 / 08	$8C57	IBM-34 Diskette formatieren
86D9	67 8D	9 / 09	$8D67	IBM-34 Sektor lesen
86DC	F6 8D	10 / 0A	$8DF6	IBM-34 Sektor schreiben
86DE	C6 8E	11 / 0B	$8EC6	IBM-34 Sektor verify
86E0	18 8F	12 / 0C	$8F18	IBM-34 Sektor auf Leerbyte prüfen
86E2	5F 8F	13 / 0D	$8F5F	IBM-34 Sektorfolge ermitteln
86E4	B3 89	14 / 0E	$89B3	IBM-34 Spur initialisieren

[839D/8408/8499/84C5/8520/BF4E]

Routine um IBM-34 Funktionen aufzurufen (Nummer im Akku)

```
86E6    78         SEI          Bus- und Kontrolleraufruf abschalten
86E7    48         PHA          Nummer der aufzurufenden Routine merken
86E8    AA         TAX          und entsprechendes
86E9    BD B9 86   LDA $86B9,X  Steuerbyte der Routine holen
86EC    85 1B      STA $1B      und speichern
86EE    A5 5E      LDA $5E      Flag für IBM-34 Format
86F0    09 80      ORA #$80     in Befehls-Statusbyte
86F2    85 5E      STA $5E      setzen
86F4    06 1B      ASL $1B      Bit7 des Steuerbytes prüfen
86F6    90 05      BCC $86FD    gesetzt ?
86F8    AD 03 02   LDA $0203    ja, Nummer der anzusteuernden Spur
86FB    85 67      STA $67      holen und Zeiger setzen
86FD[1] 06 1B      ASL $1B      Bit6 des Steuerbytes prüfen
86FF    90 05      BCC $8706    gesetzt ?
8701    AD 04 02   LDA $0204    ja, Nummer des gewünschten Sektors
8704    85 43      STA $43      holen und merken
8706[1] 06 1B      ASL $1B      Bit5 des Steuerbytes prüfen
8708    90 11      BCC $871B    gesetzt ?
870A    AD 00 1C   LDA $1C00    Bussteuerregister holen und
870D    29 10      AND #$10     Bit für 'Write Protect' testen
870F    D0 0A      BNE $871B    ist ein Schreibschutz vorhanden ?
8711    A5 3B      LDA $3B      ja, Flag für
8713    09 08      ORA #$08     'Schreibschutz angebracht'
8715    85 3B      STA $3B      setzen
8717    A2 08      LDX #$08     Fehlernummer für 'Write Protect On'
8719    86 46      STX $46      als auszugebendes Zeichen merken
871B[2] 06 1B      ASL $1B      Bit4 des Steuerbytes prüfen
871D    90 03      BCC $8722    gesetzt ?
871F    20 94 87   JSR $8794    ja, Laufwerks-Motor anschalten
8722[1] 06 1B      ASL $1B      Bit3 des Steuerbytes prüfen
8724    90 03      BCC $8729    gesetzt ?
8726    20 BA 87   JSR $87BA    ja, Zielspur aus ($67) ansteuern
8729[1] 06 1B      ASL $1B      Bit2 des Steuerbytes prüfen
872B    90 03      BCC $8730    gesetzt ?
872D    20 B0 87   JSR $87B0    warten bis Kopf und Motor bereit sind
8730[1] 20 54 89   JSR $8954    Kopf auf aktueller Seite aktivieren
8733    06 1B      ASL $1B      Bit1 des Steuerbytes prüfen
8735    90 03      BCC $873A    gesetzt ?
8737    20 2A 89   JSR $892A    ja, IBM-34 Header lesen und Kopf setzen
```

```
873A[1] A9 00     LDA #$00       Prozessor-Statusregister löschen
873C    68        PLA            Nummer des aufzurufenden Programms
873D    0A        ASL A          holen und verdoppeln (da Adress-Tabelle
873E    AA        TAX            aus 2-Byte-Zeigern besteht)
873F    BD C8 86  LDA $86C8,X    Programm-Adresse (Low-Byte) aus Tabelle
8742    85 6F     STA $6F        ermitteln und in Zeiger setzen
8744    BD C9 86  LDA $86C9,X    High-Byte holen und
8747    85 70     STA $70        merken
8749    20 61 87  JSR $8761      Programm ausführen
874C    20 8F F9  JSR $F98F      Motor ausschalten (Flag setzen)
874F    AE B0 01  LDX $01B0      Rückmeldung des letzten Jobs holen
8752    E0 02     CPX #$02       und mit Wert für 'Ok' vergleichen
8754    08        PHP            Ergebnis merken
8755    06 1B     ASL $1B        Bit0 des Steuerbytes prüfen
8757    B0 06     BCS $875F      gesetzt ?
8759    28        PLP            nein, Ergebnis der Fehlerprüfung holen
875A    90 04     BCC $8760      ist Job fehlerfrei ausgeführt worden ?
875C    4C 8C 83  JMP $838C      nein, Fehler ausgeben und Ende
875F[1] 28        PLP            Ergebnis der Fehlerprüfung bereitstellen
8760[1] 60        RTS            zurück zur aufrufenden Routine
8761[1] 6C 6F 00  JMP ($006F)    IBM-34 Routine ausführen
--------------------------------------------------------------------------
[87A3/99E7/A642/BF51]
Laufwerksmotor einschalten
8764    08        PHP            Prozessorstatus retten
8765    78        SEI            Bus- und Kontrolleraufruf abschalten
8766    AD 00 1C  LDA $1C00      Bussteuerregister holen
8769    09 04     ORA #$04       und Bit für 'Motor an' setzen
876B    8D 00 1C  STA $1C00      Register wieder speichern
876E    28        PLP            Prozessorstatus wieder herstellen
876F    60        RTS            zurück zur aufrufenden Routine
--------------------------------------------------------------------------
[99FB/9A39/A654/BF54]
Laufwerksmotor ausschalten
8770    08        PHP            Prozessorstatus retten
8771    78        SEI            Bus- und Kontrolleraufruf abschalten
8772    AD 00 1C  LDA $1C00      Bussteuerregister holen und
8775    29 FB     AND #$FB       Bit für 'Motor an' löschen
8777    8D 00 1C  STA $1C00      Steuerregister wieder setzen
877A    28        PLP            Prozessorstatus wieder herstellen
877B    60        RTS            zurück zur aufrufenden Routine
```

```
---------------------------------------------------------------------------
[884F]  vgl. C100/C118
LED am Laufwerk einschalten
877C  08          PHP             Prozessorstatus retten
877D  78          SEI             Bus- und Kontrolleraufruf abschalten
877E  AD 00 1C    LDA $1C00       Bussteuerregister holen
8781  09 08       ORA #$08        und Bit für 'LED an' setzen
8783  8D 00 1C    STA $1C00       Register wieder speichern
8786  28          PLP             Prozessorstatus wieder herstellen
8787  60          RTS             zurück zur aufrufenden Routine
---------------------------------------------------------------------------
[8861]
LED am Laufwerk ausschalten
8788  08          PHP             Prozessorstatus retten
8789  78          SEI             Bus- und Kontrolleraufruf abschalten
878A  AD 00 1C    LDA $1C00       Bussteuerregister holen
878D  29 F7       AND #$F7        und Bit für 'LED an' löschen
878F  8D 00 1C    STA $1C00       Steuerregister wieder setzen
8792  28          PLP             Prozessorstatus wieder herstellen
8793  60          RTS             zurück zur aufrufenden Routine
---------------------------------------------------------------------------
[871F]
Motor anschalten und Flags initialisieren
8794  08          PHP             Prozessorstatus retten
8795  78          SEI             Bus- und Kontrolleraufruf abschalten
8796  A5 20       LDA $20         Laufwerksstatus holen
8798  C9 20       CMP #$20        und auf 'Motor läuft' prüfen
879A  F0 0E       BEQ $87AA       ist der Motor schon aktiv ?
879C  AD 02 02    LDA $0202       nein, Jobcode der Routine holen
879F  29 01       AND #$01        und gewünschtes Laufwerk feststellen
87A1  85 3E       STA $3E         Nummer als aktuelles Laufwerk setzen
87A3  20 64 87    JSR $8764       Motor anschalten
87A6  A9 A0       LDA #$A0        Laufwerksstatus auf
87A8  85 20       STA $20         'Motor an/noch nicht auf Drehzahl'
87AA  A9 32       LDA #$32        Zähler für Hochlaufzeit des
87AC  85 48       STA $48         Motors setzen
87AE  28          PLP             Statusregister wieder holen
87AF  60          RTS             zurück zur aufrufenden Routine
---------------------------------------------------------------------------
```

```
[872D]
Warten bis Motor auf Drehzahl ist und Kopf auf Position steht
87B0  08        PHP         Status retten
87B1  58        CLI         Bus- und Kontrolleraufruf einschalten
87B2  A5 20     LDA $20     Laufwerksstatus holen und auf
87B4  C9 20     CMP #$20    'kein Stepper / Laufwerk bereit' prüfen
87B6  D0 FA     BNE $87B2   Laufwerk bereit ?
87B8  28        PLP         ja, Status wieder herstellen
87B9  60        RTS         zurück zur aufrufenden Routine
-----------------------------------------------------------------
```

```
[85E5/8726/8927/894C/89FA/8D0A/8D3A/8D51/8F6D]
Neue Spur ansteuern
87BA  08        PHP         Status retten
87BB  58        CLI         Bus- und Kontrolleraufruf einschalten
87BC  A5 67     LDA $67     Nummer der neuen Spur holen und Zahl
87BE  0A        ASL A       der absoluten Halbschritte errechnen
87BF  C5 64     CMP $64     mit aktueller Kopfposition vergleichen
87C1  F0 1A     BEQ $87DD   identisch ?
87C3  A5 67     LDA $67     nein, Nummer der neuen Spur holen und
87C5  0A        ASL A       und Halbschritte errechnen
87C6  C5 64     CMP $64     mit aktuellem Zählerstand vergleichen
87C8  F0 0E     BEQ $87D8   identisch ?
87CA  B0 06     BCS $87D2   nein, ist akt. Zähler größer Zielpos. ?
87CC  20 E7 87  JSR $87E7   ja, einen Halbschritt nach außen
87CF  4C C3 87  JMP $87C3   weiter bis Spur erreicht
87D2  20 DF 87  JSR $87DF   einen Halbschritt nach außen
87D5  4C C3 87  JMP $87C3   weiter bis Spur erreicht
87D8  A0 10     LDY #$10    Zähler initialisieren
87DA  20 29 88  JSR $8829   40/20 ms Verzögerung (1/2 MHz Takt)
87DD  28        PLP         Status wieder herstellen
87DE  60        RTS         zurück zur aufrufenden Routine
-----------------------------------------------------------------
```

```
[87D2]
einen Halbschritt nach innen
87DF  A5 64     LDA $64     aktuelle Position feststellen
87E1  18        CLC         Addition vorbereiten
87E2  69 01     ADC #$01    einen Halbschritt addieren
87E4  4C 14 88  JMP $8814   Steppermotor ansteuern
-----------------------------------------------------------------
```

```
[87E7]  vgl. 9A66/FF45
einen Halbschritt nach außen
87E7    A0 63      LDY #$63      Zahl der Abtastversuche für Spur0 (99)
87E9[1] AD 0F 18   LDA $180F     Steuerregister A holen
87EC    6A         ROR A         Spur0-Kennung (bit0) ins Carry
87ED    08         PHP           und Carry merken
87EE    AD 0F 18   LDA $180F     Steuerregister nochmal lesen
87F1    6A         ROR A         Spur0-Kennung (Bit0)
87F2    6A         ROR A         nach Bit7 schieben
87F3    28         PLP           vorheriges Abtastergebnis holen
87F4    29 80      AND #$80      letztes Abtastergenis isolieren
87F6    90 04      BCC $87FC     ist im ersten Test Spur0 aktiv (Bit=0) ?
87F8    10 15      BPL $880F     nein, ist Spur0 jetzt erreicht ?
87FA    30 02      BMI $87FE     ja, immer Sprung nach $877C
87FC[1] 30 11      BMI $880F     Spur0 immer noch aktiv ?
------------------------------------------------------------------------
Zustand der Spur0 Lichtschranke hat sich nicht geändert
87FE[1] 88         DEY           ja, noch ein Versuch
87FF    D0 E8      BNE $87E9     schon alle Versuche durchgeführt ?
8801    B0 0C      BCS $880F     ja, ist Kopf auf Spur0-Position ?
8803    AD 00 1C   LDA $1C00     ja, Steuerregister für Steppermotor
8806    29 03      AND #$03      Stepperbits isolieren
8808    D0 05      BNE $880F     ist eine Stepperspule angesteuert ?
880A    A9 00      LDA #$00      aktuelle Kopfposition
880C    85 64      STA $64       löschen
880E    60         RTS           zurück zur aufrufenden Routine
------------------------------------------------------------------------
Zustand der Spur0 Lichtschranke hat sich geändert
880F[3] A5 64      LDA $64       aktuelle Kopfposition holen
8811    38         SEC           und um einen
8812    E9 01      SBC #$01      Halbschritt vermindern
8814[1] 85 64      STA $64       neue Position merken
8816    29 03      AND #$03      Steuerbits für Stepperspulen herstellen
8818    85 6F      STA $6F       und merken
881A    08         PHP           Prozessorstatus retten
881B    78         SEI           Bus- und Konrolleraufruf abschalten
881C    AD 00 1C   LDA $1C00     Steuerregister holen
881F    29 FC      AND #$FC      Steppersteuerung ausblenden und
8821    05 6F      ORA $6F       neue Bitwerte setzen
8823    8D 00 1C   STA $1C00     neuen Wert ins Steuerregister
8826    28         PLP           Prozessorstatus wieder herstellen
```

```
8827   A0 06      LDY #$06      Zähler für 13/7.5 ms Verzögerung setzen
8829[2] 20 30 88  JSR $8830     etwa 2.6/1.3 ms Verzögerung
882C   88         DEY           Zähler korregieren
882D   D0 FA      BNE $8829     Verzögerung schon abgelaufen ?
882F   60         RTS           ja, zurück zur aufrufenden Routine
-------------------------------------------------------------------

[8829]
etwa 2.6/1.3 ms Verzögerung (2583 Zyklen bis zur Fortsetzungsstelle)
8830   A2 02      LDX #$02      Zahl der Zählschleifen
8832   A9 00      LDA #$00      Zähler initialisieren
8834[2] 69 01     ADC #$01      und eins hochzählen
8836   D0 FC      BNE $8834     schon bis 256 gezählt ?
8838   CA         DEX           ja, nächste Zählschleife
8839   D0 F9      BNE $8834     schon alle Schleifen ?
883B   60         RTS           ja, zurück zur aufrufenden Routine
-------------------------------------------------------------------

[8A4D/8DAE/8E9D]
Fehler von CP/M-Kontroller holen
883C   EA         NOP           Verzögerung bis Kontroller bereit ist
883D   AD 00 20   LDA $2000     Statusregister abfragen
8840   4A         LSR A         Fehlerbits für 'Record Not Found' und
8841   4A         LSR A         'CRC Error' (Bit 3 und 4) in die
8842   4A         LSR A         Positionen 0 und 1 schieben
8843   29 03      AND #$03      Fehlerbits isolieren und
8845   AA         TAX           Fehlerzeiger herstellen
8846   BD 82 8A   LDA $8A82,X   Nummer der Fehlermeldung feststellen
8849   8D B0 01   STA $01B0     und setzen
884C   AA         TAX           Fehlernummer merken
884D   60         RTS           zurück zur aufrufenden Routine
-------------------------------------------------------------------

[89C0/8A30/8A88/8D85/8E63/8EDE/8F27]
Kommando an CP/M Kontroller (WD 1770) übergeben
884E   48         PHA           Kommando merken
884F   20 7C 87   JSR $877C     LED am Laufwerk anschalten
8852   68         PLA           Kommando wieder holen
8853   8D 00 20   STA $2000     und an CP/M Controller übergeben
8856   A9 01      LDA #$01      Bit für 'Busy - Flag'
8858   EA         NOP           Verzögerung bis Kontroller bereit ist
8859[1] 2C 00 20  BIT $2000     Statusregister des Kontrollers testen
885C   F0 FB      BEQ $8859     ist Befehl übernommen (Busy gesetzt) ?
885E   4C 7E A4   JMP $A47E     ja, 45 Taktzyklen warten
```

```
------------------------------------------------------------------
[89C3/8A4A/8C48/8DAB/8F15/8F5C]
Warten bis aktueller Befehl des CP/M Kontrollers zu Ende ist
8861    20 88 87  JSR $8788     LED ausschalten
8864    A9 01     LDA #$01      Bit für 'Busy - Flag'
8866[1] 2C 00 20  BIT $2000     im Statusregister testen
8869    D0 FB     BNE $8866     ist Befehl noch aktiv ?
886B    60        RTS           nein, zurück zur aufrufenden Routine
------------------------------------------------------------------
[8DED/886C]
Nummer des nächsten Sektors errechnen
886C    A5 60     LDA $60       kleinste Sektornummer holen
886E    38        SEC           und Versatz bis
886F    E9 01     SBC #$01      zu Sektornummer Null errechnen
8871    85 46     STA $46       und merken
8873    AD 04 02  LDA $0204     Nummer des aktuellen Sektors holen
8876    18        CLC           und dazu den
8877    65 3C     ADC $3C       Sektorversatz addieren
8879    C5 61     CMP $61       mit maximaler Sektornummer vergleichen
887B    F0 07     BEQ $8884     ist neue Nummer identisch ?
887D    90 05     BCC $8884     nein, ist neue Nummer kleiner ?
887F    E5 61     SBC $61       nein, Sektornummer auf
8881    18        CLC           erlaubten Bereich umrechnen und
8882    65 46     ADC $46       allgem. Sektorverschiebung beachten
8884[2] 8D 04 02  STA $0204     neue Sektornummer setzen
8887    60        RTS           zurück zur aufrufenden Routine
------------------------------------------------------------------
[8CF9]
Tabelle der Sektornummern für Formatieren erstellen
8888    A0 00     LDY #$00      Zeiger auf aktuelle Sektornummer löschen
888A    A2 00     LDX #$00      Zähler für Sektoren löschen
888C    AD 03 02  LDA $0203     Nummer des ersten
888F    29 3F     AND #$3F      Sektors auf Bereich 0-63
8891    8D 03 02  STA $0203     beschränken
8894    85 60     STA $60       und als kleinsten Sektor merken
8896    48        PHA           Sektornummer merken
8897    AD 07 02  LDA $0207     Nummer des letzten Sektors
889A    48        PHA           retten
889B    EE 04 02  INC $0204     Sektorversatz herstellen
889E[1] AD 03 02  LDA $0203     Nummer des aktuellen Sektors holen und
88A1    99 0B 02  STA $020B,Y   in Tabelle eintragen
```

```
88A4    EE 03 02   INC $0203    Nummer des nächsten Sektors setzen
88A7    E8         INX          Zahl der angelegten Sektoren
88A8    98         TYA          Zeiger auf Sektorposition
88A9    18         CLC          holen und Sektorversatz
88AA    6D 04 02   ADC $0204    einrechnen
88AD    A8         TAY          neuen Zeiger merken und mit
88AE    C0 20      CPY #$20     max. Sektornummer vergleichen
88B0    B0 0C      BCS $88BE    wurde 32 überschritten ?
88B2    CC 07 02   CPY $0207    nein, Nummer des letzten Sektors testen
88B5    90 1A      BCC $88D1    Nummer erreicht ?
88B7    D0 12      BNE $88CB    nein, letzte Nummer erreicht ?
88B9    EC 07 02   CPX $0207    ja, Zahl der angelegten Sektoren testen
88BC    F0 0D      BEQ $88CB    sind alle Sektoren angelegt ?
88BE[2] CE 04 02   DEC $0204    nein, Sektorversatz korregieren
88C1    68         PLA          max. Sektornummer wieder
88C2    8D 07 02   STA $0207    herstellen
88C5    68         PLA          Nummer des ersten Sektors holen
88C6    8D 03 02   STA $0203    und setzen
88C9    38         SEC          Flag für 'Fehler aufgetreten'
88CA    60         RTS          zurück zur aufrufenden Routine
88CB[2] 98         TYA          Zeiger auf aktuelle Sektorposition
88CC    38         SEC          in erlaubten
88CD    ED 07 02   SBC $0207    Sektorbereich umrechnen
88D0    A8         TAY          und merken
88D1[1] EC 07 02   CPX $0207    Zahl der angelegten Sektoren testen
88D4    D0 C8      BNE $889E    schon alle Sektornummern in Tabelle ?
88D6    86 97      STX $97      ja, Zahl der Sektoren merken
88D8    CA         DEX          Nummer des letzten Sektors
88D9    8A         TXA          bilden
88DA    18         CLC          Nummer des kleinsten
88DB    65 60      ADC $60      Sektors dazurechnen und als
88DD    85 61      STA $61      Nummer des größten Sektors merken
88DF    C5 60      CMP $60      mit kleinster Nummer vergleichen
88E1    90 DB      BCC $88BE    wurde Sektor angelegt ?
88E3    68         PLA          nein, max. Sektornummer wieder
88E4    8D 07 02   STA $0207    herstellen
88E7    68         PLA          Nummer des ersten Sektors holen
88E8    8D 03 02   STA $0203    und setzen
88EB    CE 04 02   DEC $0204    Sektorversatz korregieren
88EE    18         CLC          Flag für 'kein Fehler aufgetreten'
88EF    60         RTS          zurück zur aufrufenden Routine
```

```
-----------------------------------------------------------------
[8D1F]
CP/M-Sektoren nach Formatieren auf Leerbytes überprüfen
88F0  AD B0 01   LDA $01B0      aktuelle Spur beim
88F3  48         PHA            Formatieren merken
88F4  A0 00      LDY #$00       Zähler für aktuelle Sektornummer
88F6  84 24      STY $24        löschen
88F8  A4 24      LDY $24        Nummer des aktuellen Sektors holen
88FA  B9 0B 02   LDA $020B,Y    und Sektornummer des Headers feststellen
88FD  8D 02 20   STA $2002      Sektor an CP/M-Kontroller übergeben
8900  20 18 8F   JSR $8F18      Sektor überprüfen
8903  AE B0 01   LDX $01B0      Rückmeldung holen
8906  E0 02      CPX #$02       und mit Wert für 'Ok' vergleichen
8908  B0 0B      BCS $8915      ist ein Fehler aufgetreten ?
890A  E6 24      INC $24        nein, nächsten Sektor anwählen
890C  A4 24      LDY $24        aktuelle Sektorzahl holen
890E  CC 07 02   CPY $0207      und mit maximaler Anzahl vergleichen
8911  D0 E5      BNE $88F8      schon alle Sektoren überprüft ?
8913  18         CLC            ja, Flag für 'Ok' setzen
8914  24         .byte $24      nächstes Byte überspringen (Bit-Befehl)
8915  38         SEC            Flag für Fehler setzen
8916  68         PLA            aktuelle Spurnummer des
8917  8D B0 01   STA $01B0      Formatiervorgangs wieder setzen
891A  60         RTS            zurück zur aufrufenden Routine
-----------------------------------------------------------------
[8DF3/8EC2]
nächste Spur setzen
891B  AD 74 02   LDA $0274      Länge des Befehlsstrings im
891E  C9 07      CMP #$07       Eingabepuffer auf 7 prüfen
8920  90 F8      BCC $891A      ist der Befehl kleiner 7 Zeichen ?
8922  AD 06 02   LDA $0206      nein, siebtes Zeichen holen
8925  85 67      STA $67        und als aktuelle Zielspur übernehmen
8927  4C BA 87   JMP $87BA      Spur ansteuern
-----------------------------------------------------------------
[8737]
nächsten IBM-34 Sektorheader lesen und Kopf entsprechend setzen
892A  AD B0 01   LDA $01B0      aktuelle Fehlerrückmeldung
892D  48         PHA            retten
892E  20 27 8A   JSR $8A27      nächsten IBM-34 Header lesen
8931  AE B0 01   LDX $01B0      Rückmeldung holen
8934  E0 02      CPX #$02       und auf Fehlermeldung prüfen
```

```
8936    90 0D      BCC $8945     ist Header fehlerfrei gelesen worden ?
8938    20 EF 89   JSR $89EF     ja, Kopf auf Spur 0 setzen
893B    20 27 8A   JSR $8A27     nächsten Header lesen
893E    AE B0 01   LDX $01B0     Rückmeldung holen
8941    E0 02      CPX #$02      und auf Fehlermeldung prüfen
8943    B0 0A      BCS $894F     ist Header fehlerfrei gelesen worden ?
8945[1] A5 67      LDA $67       ja, Nummer der aktuellen Zielspur holen
8947    0A         ASL A         und Zahl der Schritte bestimmen
8948    C5 64      CMP $64       mit aktuelle Position vergleichen
894A    F0 03      BEQ $894F     ist Spur schon erreicht ?
894C    20 BA 87   JSR $87BA     nein, Kopf auf Zielspur setzen
894F[2] 68         PLA           vorherige Fehlernummer wieder holen
8950    8D B0 01   STA $01B0     und setzen
8953    60         RTS           zurück zur aufrufenden Routine
------------------------------------------------------------------------
[8730/8CD5]
Kopf auf aktueller Diskettenseite aktivieren
8954    08         PHP           Prozessorstatus retten
8955    78         SEI           Bus- und Kontrolleraufruf abschalten
8956    A5 3B      LDA $3B       Flag aus Befehlsnummer
8958    29 10      AND #$10      holen
895A    C9 10      CMP #$10      Flag (Bit4) in Carry übernehmen
895C    20 F3 93   JSR $93F3     Kopf auf angewählten Seite ansteuern
895F    28         PLP           Prozessorstatus wieder herstellen
8960    60         RTS           zurück zur aufrufenden Routine
------------------------------------------------------------------------
[852A]
kleinste und größte Sektornummer feststellen
8961    A4 97      LDY $97       Zahl der angelegten Sektoren
8963    88         DEY           Zähler auf letzte Sektorposition
8964    A9 FF      LDA #$FF      max. möglicher Nummernwert
8966[1] D9 0B 02   CMP $020B,Y   mit Sektornummer vergleichen
8969    90 03      BCC $896E     ist Sektornummer kleiner ?
896B    B9 0B 02   LDA $020B,Y   ja, neue Sektornummer holen
896E[1] 88         DEY           und Zeiger auf nächste Sektornennung
896F    10 F5      BPL $8966     schon alle Sektoren geprüft ?
8971    85 60      STA $60       ja, kleinste Sektornummer setzen
8973    A4 97      LDY $97       Zahl der angelegten Sektoren
8975    88         DEY           Zähler auf letzte Sektorposition
8976    A9 00      LDA #$00      kleinster Wert
8978[1] D9 0B 02   CMP $020B,Y   mit Sektornummer vergleichen ?
```

```
897B    B0 03      BCS $8980       ist Nummer größer ?
897D    B9 0B 02   LDA $020B,Y     ja, neue Sektornummer übernehmen
8980[1] 88         DEY             Zeiger auf nächste Sektornennung
8981    10 F5      BPL $8978       schon alle Sektoren geprüft ?
8983    85 61      STA $61         ja, größte Sektornummer merken
8985    60         RTS             zurück zur aufrufenden Routine
-----------------------------------------------------------------------
```

[852D]

Sektorversatz aus Sektorfolge errechnen

```
8986    A6 97      LDX $97         Zahl der Sektoren in Tabelle
8988    A0 00      LDY #$00        Positionszeiger zurücksetzen
898A[1] B9 0B 02   LDA $020B,Y     Sektornummer aus Tabelle holen
898D    C5 60      CMP $60         und mit kleinster Nummer vergleichen
898F    F0 05      BEQ $8996       identisch ?
8991    C8         INY             nein, Zeiger auf nächsten Sektor
8992    C4 97      CPY $97         mit Zahl der Sektornummern vergleichen
8994    D0 F4      BNE $898A       schon alle geprüft ?
8996[1] 84 5F      STY $5F         ja, Stelle des kleinsten Sektors merken
8998    A5 60      LDA $60         kleinste Sektornummer
899A    18         CLC             holen und Nummer des nächsten Sektors
899B    69 01      ADC #$01        bilden
899D    85 46      STA $46         Nummer merken
899F    A2 FF      LDX #$FF        Zähler für Sektorversatz initialisieren
89A1[2] B9 0B 02   LDA $020B,Y     Sektornummer aus Tabelle holen und mit
89A4    C5 46      CMP $46         zweitem Sektor vergleichen
89A6    F0 0A      BEQ $89B2       identisch ?
89A8    E8         INX             nein, Sektorversatz erhöhen
89A9    C8         INY             Zeiger auf nächste Sektornummer
89AA    C4 97      CPY $97         auf Zahl der Sektoren prüfen
89AC    D0 F3      BNE $89A1       schon alle Sektoren bearbeitet ?
89AE    A0 00      LDY #$00        ja, Zeiger zurücksetzen
89B0    F0 EF      BEQ $89A1       immer Sprung nach $89A1
89B2[1] 60         RTS             zurück zur aufrufenden Routine
-----------------------------------------------------------------------
```

[8A0C/8CDE]

CP/M Kontroller auf Spur initialisieren

```
89B3    A5 6F      LDA $6F         Zeropagestelle retten, da als
89B5    48         PHA             Zwischenspeicher verwendet
89B6    08         PHP             Prozessorstatus merken
89B7    78         SEI             Bus- und Kontrolleraufruf abschalten
89B8    AD 01 20   LDA $2001       aktuelle Spurnummer als neuen zu
```

```
89BB  8D 03 20   STA $2003       initialisierende Spur setzen
89BE  A9 18      LDA #$18        %00011000 'Seek' (Spur setzen)
89C0  20 4E 88   JSR $884E       Kommando an CP/M-Kontroller
89C3  20 61 88   JSR $8861       warten bis Befehl ausgeführt
89C6  A2 00      LDX #$00        Zähler für Zahl der
89C8  A0 80      LDY #$80        Versuche löschen
89CA  AD 00 20   LDA $2000       CP/M-Statusregister lesen
89CD  29 02      AND #$02        Flag für Zustand des Indexlochs holen
89CF  85 6F      STA $6F         und merken
89D1  AD 00 20   LDA $2000       Statusregister nochmals lesen
89D4  29 02      AND #$02        und Zustand der Indexlochlichtschranke
89D6  C5 6F      CMP $6F         mit vorherigem vergleichen
89D8  F0 04      BEQ $89DE       ist Indexloch aufgetreten ?
89DA  28         PLP             ja, Prozessorstatus wieder herstellen
89DB  4C E7 89   JMP $89E7       Index-Flag setzen und Ende
89DE  CA         DEX             Zähler für Versuche (Low-Byte)
89DF  D0 F0      BNE $89D1       ist Zähler abglaufen ?
89E1  88         DEY             ja, High-Byte erniedrigen
89E2  D0 ED      BNE $89D1       ist Zähler abgelaufen
89E4  28         PLP             ja, Prozessorstatus wieder herstellen
89E5  38         SEC             Flag für 'Indexloch nicht aufgetreten'
89E6  24 18      .byte $24       nächstes Byte überspringen (Bit-Befehl)
89E7  18         CLC             Flag für 'Indexloch aufgetreten'
89E8  68         PLA             Zeropagestelle
89E9  85 6F      STA $6F         wieder einrichten
89EB  60         RTS             zurück zur aufrufenden Routine
-----------------------------------------------------------------
[Vektor: 86C8]
89EC  4C A0 EA   JMP $EAA0       1571-Reset ausführen
-----------------------------------------------------------------
[8938/8A09/8CDB/8CE8/8F61/Vektor: 86CA]
Kopf auf Spur0 zurückstellen
89EF  A9 B4      LDA #$B4        Zahl der aktuellen Halbspurschritte für
89F1  85 64      STA $64         Spur 37 setzen
89F3  A9 00      LDA #$00        CP/M-Kontroller auf
89F5  8D 01 20   STA $2001       Spur 0 einstellen
89F8  85 67      STA $67         neue Zielspur setzen
89FA  4C BA 87   JMP $87BA       Kopf positionieren
-----------------------------------------------------------------
```

```
[Vektor: 86CC]
Zustand der Schreibschutz-Lichtschranke (Write Protect) prüfen
89FD  AD 00 1C   LDA $1C00      Laufwerkssteuerregister holen und Bit
8A00  29 10      AND #$10       für 'Write Protect' holen (Low aktiv)
8A02  60         RTS            zurück zur aufrufenden Routine
-----------------------------------------------------------------
[Vektor: 86CF]
Spurparameter setzen
8A03  84 67      STY $67        anzusteuernde Spur setzen
8A05  86 64      STX $64        aktuelle Position in Halbspurschritten
8A07  60         RTS            zurück zur aufrufenden Routine
-----------------------------------------------------------------
[Vektor: 864E]
8A08  60         RTS            zurück zur aufrufenden Routine
-----------------------------------------------------------------
[Vektor: 86D2]
Header des nächsten CP/M Sektors lesen und in Puffer $0024
8A09  20 EF 89   JSR $89EF      Kopf auf Spur0 setzen
8A0C  20 B3 89   JSR $89B3      Kontroller initialisieren
8A0F  B0 0F      BCS $8A20      Indexloch vorhanden ?
8A11  20 27 8A   JSR $8A27      ja, Header lesen und Zeiger setzen
8A14  BD 7E 8A   LDA $8A7E,X    Zahl der Sektoren einer Spur holen
8A17  85 97      STA $97        und merken
8A19  85 61      STA $61        als größte Sektornummer setzen
8A1B  A9 01      LDA #$01       kleinste Sektornummer
8A1D  85 60      STA $60        festlegen
8A1F  60         RTS            zurück zur aufrufenden Routine
8A20[1] A9 0D    LDA #$0D       Fehlermeldung
8A22  8D B0 01   STA $01B0      'Index Not Found' setzen
8A25  D0 3E      BNE $8A65      immer Sprung nach $8A65
-----------------------------------------------------------------
[892E/893B/8A11/8F74/8F82]
nächsten IBM-System-34 Header lesen und Sektorzeiger setzen
8A27  A9 00      LDA #$00       Zeiger für
8A29  8D 71 02   STA $0271      Zahl der Bytes pro Sektorabschnitt und
8A2C  85 44      STA $44        Zahl der Abschnitte löschen
8A2E  A9 C8      LDA #$C8       %11001000 'Read Adress' (Header lesen)
8A30  20 4E 88   JSR $884E      Kommando an CP/M-Kontroller
8A33  A2 00      LDX #$00       Pufferzeiger löschen
8A35  A0 06      LDY #$06       Zahl der Headerbytes
8A37[2] AD 00 20 LDA $2000      Statusregister lesen
```

```
8A3A  29 03     AND #$03      und Flag isolieren
8A3C  4A        LSR A         Flag für 'Kommando in Ausführung' (Busy)
8A3D  90 0B     BCC $8A4A     ist Kommando noch aktiv ?
8A3F  F0 F6     BEQ $8A37     ja, weitere Headerdaten vorhanden ?
8A41  AD 03 20  LDA $2003     ja, Datenbyte holen
8A44  95 24     STA $24,X     und in Headerpuffer schreiben
8A46  E8        INX           Pufferzeiger auf nächstes Byte setzen
8A47  88        DEY           Zahl der Headerbytes erniedrigen
8A48  D0 ED     BNE $8A37     schon alle Bytes gelesen ?
8A4A  20 61 88  JSR $8861     ja, warten bis Kommado beendet ist
8A4D  20 3C 88  JSR $883C     Rückmeldung vom Kontroller holen
8A50  A5 24     LDA $24       Spurnummer aus gelesenem Header holen
8A52  0A        ASL A         und Zahl der Halbschritte errechnen
8A53  85 64     STA $64       als aktuelle Kopfposition merken
8A55  A5 27     LDA $27       Kennzeichen für Sektorgröße holen
-----------------------------------------------------------------
[8C7B/8C9F]
Zeiger für Sektortyp setzen
8A57  29 03     AND #$03      signifikante Bits isolieren
8A59  AA        TAX           und Wert merken
8A5A  BD 72 8A  LDA $8A72,X   Zahl der Bytes pro Sektorabschnitt holen
8A5D  8D 71 02  STA $0271     und merken
8A60  BD 76 8A  LDA $8A76,X   Zahl der Abschnitte pro Sektor
8A63  85 44     STA $44       feststellen und übernehmen
8A65  A5 5E     LDA $5E       Befehls-Statusbyte holen
8A67  29 80     AND #$80      und Flag für IBM-34 Diskette isolieren
8A69  0D B0 01  ORA $01B0     Nummer der aktuellen Spur einblenden
8A6C  1D 7A 8A  ORA $8A7A,X   und Kennzeichen für Sektorlänge setzen
8A6F  85 5E     STA $5E       Befehls-Statusbyte wieder setzen
8A71  60        RTS           zurück zur aufrufenden Routine
-----------------------------------------------------------------
[8A5A]  Zahl der Bytes pro Sektorabschnitt
8A72  7F                      Wert für  128 Bytes / Sektor
8A73  FF                      Wert für  256 Bytes / Sektor
8A74  FF                      Wert für  512 Bytes / Sektor
8A75  FF                      Wert für 1024 Bytes / Sektor
-----------------------------------------------------------------
```

```
[8A60]  Zahl der Sektorabschnitte pro CP/M-Sektor
8A76  01                     Wert für  128 Bytes / Sektor
8A77  01                     Wert für  256 Bytes / Sektor
8A78  02                     Wert für  512 Bytes / Sektor
8A79  04                     Wert für 1024 Bytes / Sektor
-----------------------------------------------------------------
[8A6C]  Kennzeichen für Sektorlänge (in höherwertiger Bytehälfte)
8A7A  00                     Wert für  128 Bytes / Sektor
8A7B  10                     Wert für  256 Bytes / Sektor
8A7C  20                     Wert für  512 Bytes / Sektor
8A7D  30                     Wert für 1024 Bytes / Sektor
-----------------------------------------------------------------
[8A14]  Zahl der Sektoren pro Spur; Nummer des größten Sektors
8A7E  1A                     Wert für  128 Bytes / Sektor
8A7F  10                     Wert für  256 Bytes / Sektor
8A80  09                     Wert für  512 Bytes / Sektor
8A81  05                     Wert für 1024 Bytes / Sektor
-----------------------------------------------------------------
[8846]  CP/M-Fehlermeldungen
8A82  01                     Nummer für 'Ok'
8A83  09                     Nummer für 'Prüfsumme falsch'
8A84  02                     Nummer für 'Sektorheader nicht gefunden'
8A85  03                     Nummer für 'Sync nicht gefunden'
-----------------------------------------------------------------
[8D14]
CP/M-Spur im 'IBM System 34 - Format' formatieren
8A86  A9 F8     LDA #$F8     %111110000 'Write Track' Spur schreiben
8A88  20 D0 87  JSR $87D0    Kommando an CP/M-Kontroller übergeben
8A8B  24 3B     BIT $3B      Flag in Befehlsnummer testen
8A8D  50 62     BVC $8AF1    soll Spurindex geschrieben werden ?
-----------------------------------------------------------------
Spur-Index-Marke schreiben (nach Indexloch)
8A8F  A2 50     LDX #$50     ja, Zahl der Bytes für Index-Puls (80)
8A91  AD 00 20  LDA $2000    Statusregister holen
8A94  29 03     AND #$03     und Kommandobits isolieren
8A96  4A        LSR A        Bit für 'Busy' prüfen
8A97  90 60     BCC $8AF9    wird Kommando noch ausgeführt ?
8A99  F0 F6     BEQ $8A91    ja, Kontroller für Daten bereit ?
8A9B  A9 4E     LDA #$4E     Bytewert für Pre-Index 1
8A9D  8D 03 20  STA $2003    auf Diskette schreiben
8AA0  CA        DEX          nächstes Byte schreiben
```

```
8AA1     D0 EE      BNE $8A91     schon alle Bytes ?
8AA3     A2 0C      LDX #$0C      ja, Zähler für Lücke setzen (12)
8AA5[2]  AD 00 20   LDA $2000     Statusregister holen
8AA8     29 03      AND #$03      und Kommandobits isolieren
8AAA     4A         LSR A         Bit für 'Busy' prüfen
8AAB     90 4C      BCC $8AF9     wird Kommando noch ausgeführt ?
8AAD     F0 F6      BEQ $8AA5     ja, Kontroller für Daten bereit ?
8AAF     A9 00      LDA #$00      Bytewert für Pre-Index 2
8AB1     8D 03 20   STA $2003     auf Diskette schreiben
8AB4     CA         DEX           nächstes Byte schreiben
8AB5     D0 EE      BNE $8AA5     schon alle Bytes ?
8AB7     A2 03      LDX #$03      ja, Zähler setzen
8AB9[2]  AD 00 20   LDA $2000     Statusregister holen
8ABC     29 03      AND #$03      und Kommandobits isolieren
8ABE     4A         LSR A         Bit für 'Busy' prüfen
8ABF     90 38      BCC $8AF9     wird Kommando noch ausgeführt ?
8AC1     F0 F6      BEQ $8AB9     ja, Kontroller für Daten bereit ?
8AC3     A9 F6      LDA #$F6      Wert für Taktbyte $C2
8AC5     8D 03 20   STA $2003     auf Diskette schreiben
8AC8     CA         DEX           nächstes Byte schreiben
8AC9     D0 EE      BNE $8AB9     schon alle Bytes ?
8ACB[1]  AD 00 20   LDA $2000     Statusregister holen
8ACE     29 03      AND #$03      und Kommandobits isolieren
8AD0     4A         LSR A         Bit für 'Busy' prüfen
8AD1     90 26      BCC $8AF9     wird Kommando noch ausgeführt ?
8AD3     F0 F6      BEQ $8ACB     ja, Kontroller für Daten bereit ?
8AD5     A9 FC      LDA #$FC      Bytewert für 'Adress Index Mark'
8AD7     8D 03 20   STA $2003     auf Diskette schreiben
8ADA     A2 32      LDX #$32      Zähler setzen (50)
8ADC     EA         NOP           zwei Taktzyklen Verzögerung
8ADD[2]  AD 00 20   LDA $2000     Statusregister holen
8AE0     29 03      AND #$03      und Kommandobits isolieren
8AE2     4A         LSR A         Bit für 'Busy' prüfen
8AE3     90 14      BCC $8AF9     wird Kommando noch ausgeführt ?
8AE5     F0 F6      BEQ $8ADD     ja, Kontroller für Daten bereit ?
8AE7     A9 4E      LDA #$4E      Bytewert für Post-Index
8AE9     8D 03 20   STA $2003     auf Diskette schreiben
8AEC     CA         DEX           nächstes Byte schreiben
8AED     D0 EE      BNE $8ADD     schon alle Bytes ?
8AEF     F0 14      BEQ $8B05     ja, immer Sprung nach $8B05
------------------------------------------------------------------------------
```

```
[8A8D]  Sektoren formatieren
8AF1  A2 3C     LDX #$3C    Zähler setzen (60)
8AF3  AD 00 20  LDA $2000   Statusregister holen
8AF6  29 03     AND #$03    und Kommandobits isolieren
8AF8  4A        LSR A       Bit für 'Busy' prüfen
8AF9  90 28     BCC $8B23   wird Kommando noch ausgeführt ?
8AFB  F0 F6     BEQ $8AF3   ja, Kontroller für Daten bereit ?
8AFD  A9 4E     LDA #$4E    Bytewert für Lücke 1
8AFF  8D 03 20  STA $2003   auf Diskette schreiben
8B02  CA        DEX         nächstes Byte schreiben
8B03  D0 EE     BNE $8AF3   schon alle Bytes ?
8B05  A0 01     LDY #$01    ja, Zähler für Sektoren
8B07  A2 0C     LDX #$0C    Zähler setzen
8B09  AD 00 20  LDA $2000   Statusregister holen
8B0C  29 03     AND #$03    und Kommandobits isolieren
8B0E  4A        LSR A       Bit für 'Busy' prüfen
8B0F  90 12     BCC $8B23   wird Kommando noch ausgeführt ?
8B11  F0 F6     BEQ $8B09   ja, Kontroller für Daten bereit ?
8B13  A9 00     LDA #$00    Bytewert für 2.Teil der Lücke 1
8B15  8D 03 20  STA $2003   auf Diskette schreiben
8B18  CA        DEX         nächstes Byte schreiben
8B19  D0 EE     BNE $8B09   schon alle Bytes ?
8B1B  A2 03     LDX #$03    Zähler setzen
8B1D  AD 00 20  LDA $2000   Statusregister holen
8B20  29 03     AND #$03    und Kommandobits isolieren
8B22  4A        LSR A       Bit für 'Busy' prüfen
8B23  90 57     BCC $8B7C   wird Kommando noch ausgeführt ?
8B25  F0 F6     BEQ $8B1D   ja, Kontroller für Daten bereit ?
8B27  A9 F5     LDA #$F5    Wert Taktbyte $A1
8B29  8D 03 20  STA $2003   auf Diskette schreiben
8B2C  CA        DEX         nächstes Byte schreiben
8B2D  D0 EE     BNE $8B1D   schon alle Bytes ?
8B2F  AD 00 20  LDA $2000   Statusregister holen
8B32  29 03     AND #$03    und Kommandobits isolieren
8B34  4A        LSR A       Bit für 'Busy' prüfen
8B35  90 45     BCC $8B7C   wird Kommando noch ausgeführt ?
8B37  F0 F6     BEQ $8B2F   ja, Kontroller für Daten bereit ?
8B39  A9 FE     LDA #$FE    Bytewert für 'ID Adress Mark'
8B3B  8D 03 20  STA $2003   auf Diskette schreiben
8B3E  AD 00 20  LDA $2000   Statusregister holen
8B41  29 03     AND #$03    und Kommandobits isolieren
```

8B43	4A	LSR A	Bit für 'Busy' prüfen
8B44	90 36	BCC $8B7C	wird Kommando noch ausgeführt ?
8B46	F0 F6	BEQ $8B3E	ja, Kontroller für Daten bereit ?
8B48	AD B0 01	LDA $01B0	Nummer der aktuellen Spur
8B4B	8D 03 20	STA $2003	auf Diskette schreiben
8B4E [1]	AD 00 20	LDA $2000	Statusregister holen
8B51	29 03	AND #$03	und Kommandobits isolieren
8B53	4A	LSR A	Bit für 'Busy' prüfen
8B54	90 26	BCC $8B7C	wird Kommando noch ausgeführt ?
8B56	F0 F6	BEQ $8B4E	ja, Kontroller für Daten bereit ?
8B58	A5 3B	LDA $3B	Flag für aktuelle Diskettenseite
8B5A	29 10	AND #$10	holen und prüfen
8B5C	D0 03	BNE $8B61	ist Seite 1 aktiv ?
8B5E	A9 00	LDA #$00	ja, dann Kennzeichen für Seite setzen
8B60	2C	.byte $2C	nächste 2 Bytes überspringen (Bitbefehl)
8B61 [1]	A9 01	LDA #$01	Kennzeichen für Seite 2
8B63	8D 03 20	STA $2003	auf Diskette schreiben
8B66 [1]	AD 00 20	LDA $2000	Statusregister holen
8B69	29 03	AND #$03	und Kommandobits isolieren
8B6B	4A	LSR A	Bit für 'Busy' prüfen
8B6C	90 0E	BCC $8B7C	wird Kommando noch ausgeführt ?
8B6E	F0 F6	BEQ $8B66	ja, Kontroller für Daten bereit ?
8B70	B9 0A 02	LDA $020A,Y	Sektornummer
8B73	8D 03 20	STA $2003	auf Diskette schreiben
8B76 [1]	AD 00 20	LDA $2000	Statusregister holen
8B79	29 03	AND #$03	und Kommandobits isolieren
8B7B	4A	LSR A	Bit für 'Busy' prüfen
8B7C [5]	90 33	BCC $8BB1	wird Kommando noch ausgeführt ?
8B7E	F0 F6	BEQ $8B76	ja, Kontroller für Daten bereit ?
8B80	AD 05 02	LDA $0205	Kennzeichen für Sektorlänge
8B83	8D 03 20	STA $2003	auf Diskette schreiben
8B86 [1]	AD 00 20	LDA $2000	Statusregister holen
8B89	29 03	AND #$03	und Kommandobits isolieren
8B8B	4A	LSR A	Bit für 'Busy' prüfen
8B8C	90 23	BCC $8BB1	wird Kommando noch ausgeführt ?
8B8E	F0 F6	BEQ $8B86	ja, Kontroller für Daten bereit ?
8B90	A9 F7	LDA #$F7	Bytewert für 2 CRC Bytes
8B92	8D 03 20	STA $2003	auf Diskette schreiben
8B95	A2 16	LDX #$16	Zähler setzen (22)
8B97 [2]	AD 00 20	LDA $2000	Statusregister holen
8B9A	29 03	AND #$03	und Kommandobits isolieren

```
8B9C   4A         LSR A        Bit für 'Busy' prüfen
8B9D   90 12      BCC $8BB1    wird Kommando noch ausgeführt ?
8B9F   F0 F6      BEQ $8B97    ja, Kontroller für Daten bereit ?
8BA1   A9 4E      LDA #$4E     Bytewert für Lücke 2
8BA3   8D 03 20   STA $2003    auf Diskette schreiben
8BA6   CA         DEX          nächstes Byte schreiben
8BA7   D0 EE      BNE $8B97    schon alle Bytes ?
8BA9   A2 0C      LDX #$0C     Zähler setzen (12)
8BAB 2 AD 00 20   LDA $2000    Statusregister holen
8BAE   29 03      AND #$03     und Kommandobits isolieren
8BB0   4A         LSR A        Bit für 'Busy' prüfen
8BB1 3 90 38      BCC $8BEB    wird Kommando noch ausgeführt ?
8BB3   F0 F6      BEQ $8BAB    ja, Kontroller für Daten bereit ?
8BB5   A9 00      LDA #$00     Bytewert für 2.Teil der Lücke 2
8BB7   8D 03 20   STA $2003    auf Diskette schreiben
8BBA   CA         DEX          nächstes Byte schreiben
8BBB   D0 EE      BNE $8BAB    schon alle Bytes ?
8BBD   A2 03      LDX #$03     Zähler setzen
8BBF 2 AD 00 20   LDA $2000    Statusregister holen
8BC2   29 03      AND #$03     und Kommandobits isolieren
8BC4   4A         LSR A        Bit für 'Busy' prüfen
8BC5   90 24      BCC $8BEB    wird Kommando noch ausgeführt ?
8BC7   F0 F6      BEQ $8BBF    ja, Kontroller für Daten bereit ?
8BC9   A9 F5      LDA #$F5     Wert für Taktbyte $A1
8BCB   8D 03 20   STA $2003    auf Diskette schreiben
8BCE   CA         DEX          nächstes Byte schreiben
8BCF   D0 EE      BNE $8BBF    schon alle Bytes ?
8BD1 1 AD 00 20   LDA $2000    Statusregister holen
8BD4   29 03      AND #$03     und Kommandobits isolieren
8BD6   4A         LSR A        Bit für 'Busy' prüfen
8BD7   90 12      BCC $8BEB    wird Kommando noch ausgeführt ?
8BD9   F0 F6      BEQ $8BD1    ja, Kontroller für Daten bereit ?
8BDB   A9 FB      LDA #$FB     Bytewert für 'Data Adress Mark'
8BDD   8D 03 20   STA $2003    auf Diskette schreiben
8BE0   84 6F      STY $6F      aktuellen Sektorzeiger merken
8BE2   A4 44      LDY $44      Zahl der Sektorabschnitte holen
8BE4   EA         NOP          zwei Taktzyklen Verzögerung
8BE5 3 AD 00 20   LDA $2000    Statusregister holen
8BE8   29 03      AND #$03     und Kommandobits isolieren
8BEA   4A         LSR A        Bit für 'Busy' prüfen
8BEB 3 90 60      BCC $8C4D    wird Kommando noch ausgeführt ?
```

```
8BED   F0 F6      BEQ $8BE5     ja, Kontroller für Daten bereit ?
8BEF   AD 0A 02   LDA $020A     Leerbyte für Sektor
8BF2   8D 03 20   STA $2003     auf Diskette schreiben
8BF5   EC 71 02   CPX $0271     auf Länge eines Abschnittes prüfen
8BF8   F0 04      BEQ $8BFE     schon ganzer Sektorabschnitt geschrieben
8BFA   E8         INX           nein, Zähler auf nächstes Byte
8BFB   4C E5 8B   JMP $8BE5     weiter schreiben
8BFE[1] E8        INX           Zähler für Abschnittlänge initialisieren
8BFF   88         DEY           Zahl der Abschnitte erniedrigen
8C00   D0 E3      BNE $8BE5     noch Sektorabschnitte schreiben ?
8C02[1] AD 00 20  LDA $2000     nein, Statusregister holen
8C05   29 03      AND #$03      und Kommandobits isolieren
8C07   4A         LSR A         Bit für 'Busy' prüfen
8C08   90 43      BCC $8C4D     wird Kommando noch ausgeführt ?
8C0A   F0 F6      BEQ $8C02     ja, Kontroller für Daten bereit ?
8C0C   A9 F7      LDA #$F7      Bytewert für 2 CRC-Bytes
8C0E   8D 03 20   STA $2003     auf Diskette schreiben
8C11   AC 05 02   LDY $0205     Kennzeichen für Sektorlänge
8C14   B9 4F 8C   LDA $8C4F,Y   Größe der Lücke zwischen Sektoren holen
8C17   A4 6F      LDY $6F       Nummer des aktuellen Sektors
8C19   AA         TAX           Lückenzähler setzen
8C1A[2] AD 00 20  LDA $2000     Statusregister holen
8C1D   29 03      AND #$03      und Kommandobits isolieren
8C1F   4A         LSR A         Bit für 'Busy' prüfen
8C20   90 2B      BCC $8C4D     wird Kommando noch ausgeführt ?
8C22   F0 F6      BEQ $8C1A     ja, Kontroller für Daten bereit ?
8C24   A9 4E      LDA #$4E      Bytewert für Lücke 3
8C26   8D 03 20   STA $2003     auf Diskette schreiben
8C29   CA         DEX           nächstes Byte schreiben
8C2A   D0 EE      BNE $8C1A     schon alle Bytes ?
8C2C   CC 07 02   CPY $0207     Zahl der Sektoren der Spur
8C2F   F0 04      BEQ $8C35     schon alle Sektoren angelegt ?
8C31   C8         INY           nein, Sektorzähler erhöhen
8C32   4C 07 8B   JMP $8B07     nächsten Sektor schreiben
8C35[3] AD 00 20  LDA $2000     Statusregister holen
8C38   29 03      AND #$03      und Kommandobits isolieren
8C3A   4A         LSR A         Bit für 'Busy' prüfen
8C3B   90 0B      BCC $8C48     wird Kommando noch ausgeführt ?
8C3D   F0 F6      BEQ $8C35     ja, Kontroller für Daten bereit ?
8C3F   18         CLC           Bytewert
8C40   A9 4E      LDA #$4E      für Lücke 4
```

```
8C42  8D 03 20   STA $2003     auf Diskette schreiben
8C45  4C 35 8C   JMP $8C35     Rest der Spur füllen
8C48  20 61 88   JSR $8861     warten bis Kommando abgeschlossen
8C4B  18         CLC           Flag für Formatierung 'Ok' setzen
8C4C  24         .byte $24     nächstes Byte überspringen (Bit-Befehl)
8C4D  38         SEC           Flag für Formatierfehler setzen
8C4E  60         RTS           zurück zur aufrufenden Routine
-----------------------------------------------------------------------
[8C14]  Zahl der Bytes für Lücken zwischen den CP/M-Sektoren
8C4F  07                       Wert für  128 Bytes pro Sektor
8C50  0C                       Wert für  256 Bytes pro Sektor
8C51  17                       Wert für  512 Bytes pro Sektor
8C52  2C                       Wert für 1024 Bytes pro Sektor
-----------------------------------------------------------------------
[8CA7]  Zahl der CP/M Sektoren pro Spur beim Formatieren
8C53  1A                       Wert für  128 Bytes pro Sektor
8C54  10                       Wert für  256 Bytes pro Sektor
8C55  09                       Wert für  512 Bytes pro Sektor
8C56  05                       Wert für 1024 Bytes pro Sektor
-----------------------------------------------------------------------
[Vektor: 86D8]
Diskette in 'IBM System 34' formatieren
8C57  A5 3B      LDA #$3B      Flag für Schreibschutz
8C59  29 08      AND #$08      überprüfen
8C5B  F0 07      BEQ $8C64     ist 'Write Protect' gesetzt ?
8C5D  A6 46      LDX $46       ja, Fehlernummer holen
8C5F  8E B0 01   STX $01B0     und als Rückmeldung setzen
8C62  38         SEC           Flag für Fehler aufgetreten
8C63  60         RTS           zurück zur aufrufenden Routine
8C64  20 07 D3   JSR $D307     alle Kanäle löschen
8C67  AD 74 02   LDA $0274     Länge des Befehlstrings
8C6A  38         SEC           Zahl der schon
8C6B  E9 04      SBC #$04      bearbeiteten Bytes abziehen und
8C6D  A8         TAY           Wert merken
8C6E  F0 20      BEQ $8C90     sind weitere Angaben vorhanden ?
8C70  88         DEY           Zeiger auf nächstes Befehlsbyte
8C71  F0 22      BEQ $8C95     noch weitere Angaben im Befehlsstring ?
8C73  A9 00      LDA #$00      ja, erste zu formatierende Spur
8C75  8D B0 01   STA $01B0     setzen
8C78  AD 05 02   LDA $0205     Kennzeichen für Sektorlänge holen und
8C7B  20 57 8A   JSR $8A57     Zeiger entsprechend setzen
```

```
8C7E  88        DEY          Zeiger auf nächstes Befehlsbyte
8C7F  F0 21     BEQ $8CA2    noch weitere Angaben im Befehlsstring ?
8C81  88        DEY          ja, Zeiger auf nächstes Befehlsbyte
8C82  F0 23     BEQ $8CA7    noch weitere Angaben im Befehlsstring ?
8C84  88        DEY          ja, Zeiger auf nächstes Befehlsbyte
8C85  F0 26     BEQ $8CAD    noch weitere Angaben im Befehlsstring ?
8C87  88        DEY          ja, Zeiger auf nächstes Befehlsbyte
8C88  F0 2B     BEQ $8CB5    noch weitere Angaben im Befehlsstring ?
8C8A  88        DEY          ja, Zeiger auf nächstes Befehlsbyte
8C8B  F0 2D     BEQ $8CBA    noch weitere Angaben im Befehlsstring ?
8C8D  4C BF 8C  JMP $8CBF    keine Ersatz-Werte setzen
8C90  A9 00     LDA #$00     Spurangabe im Befehlsstring
8C92  8D 04 02  STA $0204    löschen
8C95  A9 00     LDA #$00     erste zu formatierende Spur
8C97  8D B0 01  STA $01B0    setzen (0)
8C9A  A9 01     LDA #$01     Kennzeichen für 256 Bytes pro
8C9C  8D 05 02  STA $0205    Sektor setzen
8C9F  20 57 8A  JSR $8A57    Sektorzeiger setzen
8CA2  A9 27     LDA #$27     größte zu formatierende Spur
8CA4  8D 06 02  STA $0206    festlegen
8CA7  BD 53 8C  LDA $8C53,X  Zahl der Sektoren pro Spur
8CAA  8D 07 02  STA $0207    feststellen und setzen
8CAD  A9 00     LDA #$00     erste logische Spurnummer
8CAF  8D 08 02  STA $0208    setzen
8CB2  8D 01 20  STA $2001    Spur an CP/M-Kontroller übergeben
8CB5  A9 00     LDA #$00     erste physikalische zu formatierende
8CB7  8D 09 02  STA $0209    Spur setzen
8CBA  A9 E5     LDA #$E5     Leerbyte zum Füllen der Sektoren
8CBC  8D 0A 02  STA $020A    merken
8CBF  20 DE 8C  JSR $8CDE    Diskettenseite IBM-Format formatieren
8CC2  AD B0 01  LDA $01B0    Rückmeldung holen
8CC5  E0 02     CPX #$02     und mit Wert für 'Ok' vergleichen
8CC7  B0 12     BCS $8CDB    ist Spur fehlerfrei formatiert ?
8CC9  A5 3B     LDA $3B      ja, Befehlsnummer holen und Flag
8CCB  29 20     AND #$20     für 'zwei Seiten' prüfen
8CCD  F0 0C     BEQ $8CDB    sollen beide Seite formatiert werden ?
8CCF  A5 3B     LDA $3B      ja, Flag für Seite 2
8CD1  09 10     ORA #$10     in Befehlsnummer
8CD3  85 3B     STA $3B      setzen
8CD5  20 54 89  JSR $8954    Kopf auf aktueller Seite aktivieren
8CD8  20 DE 8C  JSR $8CDE    Diskettenseite IBM-Format formatieren
```

```
8CDB[2] 4C EF 89   JMP $89EF       Kopf auf Spur0 setzen und Ende
------------------------------------------------------------------
[8CBF/8CD8]
Diskettenseite in 'IBM System 34' formatieren
8CDE    20 B3 89   JSR $89B3       Spur initialisieren
8CE1    B0 7C      BCS $8D5F       ist Indexloch aufgetreten ?
8CE3    A9 01      LDA #$01        ja, Flag für Diskettenwechsel (Write
8CE5    8D 0D 18   STA $180D       Protect geändert) neu initialsieren
8CE8    20 EF 89   JSR $89EF       Kopf auf Spur 0 setzen
8CEB    AD 08 02   LDA $0208       9. Zeichen aus Befehlsstring holen
8CEE    8D B0 01   STA $01B0       und als erste Spurnummer setzen
8CF1    8D 01 20   STA $2001       Spurnummer an CP/M-Kontroller übergeben
8CF4    2C 03 02   BIT $0203       4. Zeichen aus Befehlsstring holen
8CF7    70 05      BVS $8CFE       Flag für 'keine Sektortabelle' gesetzt ?
8CF9    20 88 88   JSR $8888       nein, Sektortabelle erzeugen
8CFC    B0 61      BCS $8D5F       Tabelle fehlerfrei erstellt ?
8CFE[1] AD 09 02   LDA $0209       ja, 10. Zeichen aus Befehlsstring
8D01    29 7F      AND #$7F        erste physikalische Spurnummer erstellen
8D03    F0 08      BEQ $8D0D       Kopf auf eine Startspur bewegen ?
8D05    18         CLC             ja, physikalische Spur bei
8D06    65 67      ADC $67         der Formatieren beginnen soll
8D08    85 67      STA $67         errechnen
8D0A    20 BA 87   JSR $87BA       Spur ansteuern
8D0D[2] 78         SEI             Bus- und Kontrolleraufruf abschalten
8D0E    AD 0D 18   LDA $180D       Signal von Elektronik für 'Write-Protect
8D11    4A         LSR A           hat sich geändert' überprüfen
8D12    B0 4B      BCS $8D5F       ist Diskette gewechselt worden ?
8D14    20 86 8A   JSR $8A86       nein, Spur formatieren
8D17    B0 46      BCS $8D5F       ist dabei Fehler aufgetreten ?
8D19    AD 0D 18   LDA $180D       nein, Signal von Elektronik für 'Write-
8D1C    4A         LSR A           Protect hat sich geändert' überprüfen
8D1D    B0 40      BCS $8D5F       ist Diskette gewechselt worden ?
8D1F    20 F0 88   JSR $88F0       nein, Sektoren überprüfen
8D22    B0 3B      BCS $8D5F       alle Sektoren fehlerfrei geschrieben ?
8D24    AD 0D 18   LDA $180D       ja, Signal von Elektronik für 'Write-
8D27    4A         LSR A           Protect hat sich geändert' überprüfen
8D28    B0 35      BCS $8D5F       ist Diskette gewechselt worden ?
8D2A    AD B0 01   LDA $01B0       aktuelle logische Spurnummer mit
8D2D    CD 06 02   CMP $0206       letzter Nummer vergleichen
8D30    F0 0E      BEQ $8D40       ist gewünschter Bereich formatiert ?
8D32    E6 67      INC $67         nein, nächste Zielspur setzen
```

```
8D34     EE 01 20   INC $2001      CP/M-Kontroller auf nächste Spur stellen
8D37     EE B0 01   INC $01B0      aktuelle Spurnummer erhöhen
8D3A     20 BA 87   JSR $87BA      Kopf auf Spur setzen
8D3D     4C 0D 8D   JMP $8D0D      weiter formatieren
8D40[1]  24 3B      BIT $3B        Flags in Befehlsnummer testen
8D42     10 18      BPL $8D5C      Endspur kennzeichnen / löschen ?
8D44     38         SEC            Zahl der formatierten Spuren
8D45     AD 06 02   LDA $0206      aus letzter logischer Spur
8D48     ED 08 02   SBC $0208      und erster Spurnummer errechnen
8D4B     C9 27      CMP #$27       mit max. Zahl der Spuren vergeichen
8D4D     B0 0D      BCS $8D5C      ist bis auf 2. Seite formatiert ?
8D4F     E6 67      INC $67        ja, nächste Spur auf 2. Seite
8D51     20 BA 87   JSR $87BA      ansteuern
8D54     A2 1C      LDX #$1C       7168 mal $55 (%01010101)
8D56     20 63 9D   JSR $9D63      auf Spur schreiben
8D59     20 00 FE   JSR $FE00      Kopf auf Lesen umschalten
8D5C[2]  A2 00      LDX #$00       Fehlernummer für 'Ok'
8D5E     2C         .byte $2C      nächste 2 Bytes überspringen (Bitbefehl)
8D5F[6]  A2 06      LDX #$06       Fehlernummer für 'Formatierfehler'
8D61     8E B0 01   STX $01B0      Rückmeldung setzen
8D64     4C E9 85   JMP $85E9      und für Ausgabe vorbereiten
--------------------------------------------------------------------------
[8DF0/Vektor: 86D9]
CP/M Sektor lesen und an Rechner übermitteln
8D67     A5 3B      LDA $3B        Befehlsnummer holen und Flag für
8D69     29 20      AND #$20       'nur Puffer ausgeben' testen
8D6B     D0 59      BNE $8DC6      gesetzt ?
8D6D     A9 03      LDA #$03       nein, aktuellen Pufferzeiger auf
8D6F     85 31      STA $31        Startadresse
8D71     A0 00      LDY #$00       von Puffer 0 ($0300)
8D73     84 30      STY $30        setzen
8D75     A6 44      LDX $44        Zahl der Sektorabschnitte holen
8D77     AD 03 02   LDA $0203      Spurnummer an CP/M-Kontroller
8D7A     8D 01 20   STA $2001      übergeben
8D7D     AD 04 02   LDA $0204      Nummer des gewünschten Sektors
8D80     8D 02 20   STA $2002      an CP/M-Kontroller übergeben
8D83     A9 88      LDA #$88       %10001000 'Read Sektor' (Sektor lesen)
8D85     20 4E 88   JSR $884E      Kommando an CP/M-Kontroller
8D88     EA         NOP            zwei Taktzyklen Verzögerung
8D89[3]  AD 00 20   LDA $2000      Statusregister holen
8D8C     29 03      AND #$03       und Kommandobits isolieren
```

```
8D8E  4A        LSR A          Bit für 'Busy' prüfen
8D8F  90 1A     BCC $8DAB      wird Kommando noch ausgeführt ?
8D91  29 01     AND #$01       ja, Flagbit für 'Datenregister bereit'
8D93  F0 F4     BEQ $8D89      warten bis Daten bereit sind
8D95  AD 03 20  LDA $2003      Datenbyte von CP/M-Kontroller holen
8D98  91 30     STA ($30),Y    und in Puffer schreiben
8D9A  CC 71 02  CPY $0271      Zahl der Bytes pro Sektorabschnitt
8D9D  F0 03     BEQ $8DA2      schon alle Bytes eingelesen ?
8D9F  C8        INY            nein, Pufferzeiger auf nächstes Byte
8DA0  D0 E7     BNE $8D89      Ende des Puffers erreicht ?
8DA2  C8        INY            ja, Pufferzeiger löschen
8DA3  CA        DEX            nächsten Abschnitt des Sektors
8DA4  F0 05     BEQ $8DAB      schon alle Abschnitte gelesen
8DA6  E6 31     INC $31        nein, Pufferzeiger auf nächsten Puffer
8DA8  4C 89 8D  JMP $8D89      Sektor weiter lesen
8DAB  20 61 88  JSR $8861      warten bis Kommando beendet
8DAE  20 3C 88  JSR $883C      Status des CP/M-Kontrollers holen
8DB1  20 E9 85  JSR $85E9      Fehlernummer für Ausgabe vorbereiten
8DB4  24 3B     BIT $3B        Flag für 'Fehler beachten' prüfen
8DB6  70 07     BVS $8DBF      soll Rückmeldung getestet werden ?
8DB8  E0 02     CPX #$02       ja, auf Wert für 'Ok' prüfen
8DBA  90 03     BCC $8DBF      ist Nummer größer (Fehlernummer) ?
8DBC  4C 84 83  JMP $8384      ja, Fehler ausgeben
8DBF  20 F9 85  JSR $85F9      Byte auf 1571-Bus ausgeben
8DC2  A5 3B     LDA $3B        Flag für 'nur Sektor lesen' beachten
8DC4  30 22     BMI $8DE8      soll Puffer übertragen werden ?
8DC6  A9 03     LDA #$03       ja, aktuellen Pufferzeiger $30/$31
8DC8  85 31     STA $31        auf Startadresse von
8DCA  A0 00     LDY #$00       Puffer 0 ($0300)
8DCC  84 30     STY $30        setzen
8DCE  A6 44     LDX $44        Zahl der Sektorabschnitte pro Sektor
8DD0  B1 30     LDA ($30),Y    Byte aus Puffer holen
8DD2  85 46     STA $46        und als auszugebendes Zeichen merken
8DD4  20 F9 85  JSR $85F9      Byte auf 1571-Bus ausgeben
8DD7  CC 71 02  CPY $0271      Zahl der Bytes pro Sektorabschnitt
8DDA  F0 03     BEQ $8DDF      schon ganzen Abschnitt übertragen ?
8DDC  C8        INY            nein, Pufferzeiger auf nächstes Byte
8DDD  D0 F1     BNE $8DD0      Ende des Puffers erreicht ?
8DDF  C8        INY            ja, Pufferzeiger auf Start setzen
8DE0  CA        DEX            Zahl der Sektorabschnitte erniedrigen
8DE1  F0 05     BEQ $8DE8      schon ganzer Sektor übertragen ?
```

```
8DE3     E6 31      INC $31        nein, Pufferzeiger auf nächsten Puffer
8DE5     4C D0 8D   JMP $8DD0      Daten weiter übertragen
8DE8[2]  CE 05 02   DEC $0205      Zahl der zu übertragenden Sektoren
8DEB     F0 06      BEQ $8DF3      weitere Sektoren lesen ?
8DED     20 6C 88   JSR $886C      ja, nächste Sektornummer errechnen
8DF0     4C 67 8D   JMP $8D67      nächsten Sektor lesen
8DF3[1]  4C 1B 89   JMP $891B      nächste angegebene Spur ansteuern
```

[8EBF/Vektor: 86DC]

CP/M Sektor von Rechner übertragen und auf Diskette schreiben

```
8DF6     A9 03      LDA #$03       aktuellen Pufferzeiger $30/$31
8DF8     85 31      STA $31        auf Startadresse von
8DFA     A0 00      LDY #$00       Puffer 0 ($0300)
8DFC     84 30      STY $30        setzen
8DFE     A6 44      LDX $44        Zahl der Abschnitte pro Sektor
8E00     A5 3B      LDA $3B        Flag für 'Puffer einlesen' prüfen
8E02     30 30      BMI $8E34      Daten vom Rechner übernehmen ?
8E04[2]  AD 00 18   LDA $1800      ja, Bussteuerregister holen
8E07     49 08      EOR #$08       und Clock-Ausgang umschalten
8E09     2C 0D 40   BIT $400D      Interruptregister zurücksetzen
8E0C     8D 00 18   STA $1800      Bussteuerregister setzen
8E0F[1]  AD 00 18   LDA $1800      ATN-Eingang testen
8E12     10 03      BPL $8E17      gesetzt ?
8E14     20 59 EA   JSR $EA59      auf ATN-Kommandomodus prüfen
8E17[1]  AD 0D 40   LDA $400D      nein, Flag für 'serielles
8E1A     29 08      AND #$08       Eingangsregister voll' prüfen
8E1C     F0 F1      BEQ $8E0F      sind Daten übertragen ?
8E1E     AD 0C 40   LDA $400C      ja, Byte holen
8E21     91 30      STA ($30),Y    und in Puffer schreiben
8E23     CC 71 02   CPY $0271      Zahl der Bytes pro Sektorabschnitt
8E26     F0 03      BEQ $8E2B      schon ganzer Abschnitt eingelesen ?
8E28     C8         INY            nein, Pufferzeiger auf nächstes Byte
8E29     D0 D9      BNE $8E04      Ende des Puffers erreicht ?
8E2B[1]  C8         INY            ja, Pufferzeiger auf Start setzen
8E2C     CA         DEX            nächsten Sektorabschnitt
8E2D     F0 05      BEQ $8E34      weitere Abschnitte vom Bus einlesen ?
8E2F     E6 31      INC $31        ja, Pufferzeiger auf nächsten Puffer
8E31     4C 04 8E   JMP $8E04      weiter einlesen
8E34[1]  A5 3B      LDA $3B        Befehlsnummer holen und Flag für
8E36     29 20      AND #$20       'Puffer in Sektor schreiben' testen
8E38     D0 7D      BNE $8EB7      soll Sektor geschrieben werden ?
```

```
8E3A    A5 3B      LDA $3B        ja, Flag für 'Write Protect'
8E3C    29 08      AND #$08       überprüfen
8E3E    F0 05      BEQ $8E45      ist Schreibschutz aktiv ?
8E40    A6 46      LDX $46        ja, Fehlernummer holen
8E42    4C 81 83   JMP $8381      und ausgeben
8E45[1] A9 03      LDA #$03       aktuellen Pufferzeiger $30/$31
8E47    85 31      STA $31        auf Startadresse von
8E49    A0 00      LDY #$00       Puffer 0 ($0300)
8E4B    84 30      STY $30        setzen
8E4D    A6 44      LDX $44        Zahl der Abschnitte pro Sektor
8E4F    AD 03 02   LDA $0203      Spurnummer aus Befehlsstring holen
8E52    8D 01 20   STA $2001      und an CP/M-Kontroller übergeben
8E55    AD 04 02   LDA $0204      Nummer des gewünschten Sektors holen
8E58    8D 02 20   STA $2002      und an CP/M-Kontroller übergeben
8E5B    AD 0D 18   LDA $180D      Signal von Elektronik für 'Write-Protect
8E5E    4A         LSR A          hat sich geändert'
8E5F    B0 32      BCS $8E93      wurde Diskette gewechselt ?
8E61    A9 A8      LDA #$A8       nein, %101010000 'Write single Sector'
8E63    20 4E 88   JSR $884E      Kommando an CP/M-Kontroller übermitteln
8E66[3] AD 00 20   LDA $2000      Statusregister holen
8E69    29 03      AND #$03       und Kommandobits isolieren
8E6B    4A         LSR A          Bit für 'Busy' prüfen
8E6C    90 25      BCC $8E93      wird Kommando noch ausgeführt ?
8E6E    29 01      AND #$01       ja, Flag für 'Datenregister leer'
8E70    F0 F4      BEQ $8E66      werden neue Daten angenommen ?
8E72    B1 30      LDA ($30),Y    ja, Datenbyte aus Puffer holen
8E74    8D 03 20   STA $2003      und auf Diskette schreiben
8E77    CC 71 02   CPY $0271      Zahl der Bytes pro Abschnitt
8E7A    F0 03      BEQ $8E7F      Abschnitt zu Ende ?
8E7C    C8         INY            nein, Pufferzeiger auf nächstes Byte
8E7D    D0 E7      BNE $8E66      Ende des Puffers erreicht ?
8E7F[1] C8         INY            ja, Pufferzeiger auf Start setzen
8E80    CA         DEX            Zahl der Abschnitte pro Sektor
8E81    F0 05      BEQ $8E88      noch ein Abschnitt ?
8E83    E6 31      INC $31        ja, Pufferadresse auf nächsten Puffer
8E85    4C 66 8E   JMP $8E66      weiter auf Diskette schreiben
8E88    AD 0D 18   LDA $180D      Signal von Elektronik für 'Write-Protect
8E8B    4A         LSR A          hat sich geändert' prüfen
8E8C    B0 05      BCS $8E93      wurde Diskette gewechselt ?
8E8E    20 C6 8E   JSR $8EC6      nein, Sektor zum Test einlesen
8E91    90 07      BCC $8E9A      Lesen einwandfrei funktioniert ?
```

```
8E93[3] 20 CE 81   JSR $81CE      nein, 1571 Bus auf Ausgang schalten
8E96    A2 07      LDX #$07       Fehlernummer für 'verify error'
8E98    D0 06      BNE $8EA0      immer Sprung nach $8EA0
8E9A[1] 20 CE 81   JSR $81CE      1571 Bus auf Ausgang schalten
8E9D    20 3C 88   JSR $883C      Fehlerstatus des CP/M-Kontrollers holen
8EA0[1] 8E B0 01   STX $01B0      und merken
8EA3    20 E9 85   JSR $85E9      Fehler für Ausgabe vorbereiten
8EA6    20 F9 85   JSR $85F9      Byte auf 1571-Bus ausgeben
8EA9    20 A0 86   JSR $86A0      auf Umspringen von Clock warten
8EAC    20 B2 81   JSR $81B2      1571 Bus auf Eingang schalten
8EAF    24 3B      BIT $3B        Flag für 'Fehler beachten' testen
8EB1    70 04      BVS $8EB7      soll Rückmeldung geprüft werden ?
8EB3    E0 02      CPX #$02       ja, auf Fehlernummer überprüfen
8EB5    B0 0E      BCS $8EC5      liegt ein Fehler vor ?
8EB7[2] CE 05 02   DEC $0205      nein, Zahl der zu schreibenden Sektoren
8EBA    F0 06      BEQ $8EC2      noch einen Sektor ?
8EBC    20 6C 88   JSR $886C      ja, Nummer des nächsten Sektors holen
8EBF    4C F6 8D   JMP $8DF6      nächsten Sektor einlesen und schreiben
8EC2[1] 4C 1B 89   JMP $891B      nächste Spur holen und setzen
8EC5[1] 60         RTS            zurück zur aufrufenden Routine
-------------------------------------------------------------------------
[8E8E/Vektor: 86DE]
CP/M-Sektor mit Pufferinhalt vergleichen (Verify)
8EC6    A9 03      LDA #$03       aktuellen Pufferzeiger $30/$31
8EC8    85 31      STA $31        auf Startadresse von
8ECA    A0 00      LDY #$00       Puffer 0 ($0300)
8ECC    84 30      STY $30        setzen
8ECE    A6 44      LDX $44        Zahl der Abschnitte pro Sektor
8ED0    AD 03 02   LDA $0203      Spurnummer aus Befehlsstring holen und
8ED3    8D 01 20   STA $2001      an CP/M-Kontroller übergeben
8ED6    AD 04 02   LDA $0204      Nummer des gewünschten Sektors holen
8ED9    8D 02 20   STA $2002      und an CP/M-Kontroller
8EDC    A9 88      LDA #$88       %10001000 'Read Sector' (Sektor lesen)
8EDE    20 4E 88   JSR $884E      Kommando an Kontroller übermitteln
8EE1[3] AD 00 20   LDA $2000      Statusregister holen
8EE4    29 03      AND #$03       und Kommandobits isolieren
8EE6    4A         LSR A          Bit für 'Busy' prüfen
8EE7    90 1C      BCC $8F05      wird Kommando noch ausgeführt ?
8EE9    29 01      AND #$01       ja, Flag 'für Daten bereit' prüfen
8EEB    F0 F4      BEQ $8EE1      warten bis Datenbyte fertig ist ?
8EED    AD 03 20   LDA $2003      Byte von Diskette lesen
```

```
8EF0  D1 30     CMP ($30),Y    und mit Pufferinhalt vergleichen
8EF2  D0 11     BNE $8F05      identisch ?
8EF4  CC 71 02  CPY $0271      ja, Zahl der Bytes pro Sektorabschnitt
8EF7  F0 03     BEQ $8EFC      schon ganzer Abschnitt verglichen ?
8EF9  C8        INY            nein, Pufferzeiger auf nächstes Byte
8EFA  D0 E5     BNE $8EE1      Ende des Puffers erreicht ?
8EFC[1] C8      INY            ja, Pufferzeiger auf Start setzen
8EFD  CA        DEX            Zahl der Sektorabschnitte
8EFE  F0 10     BEQ $8F10      noch ein Abschnitt ?
8F00  E6 31     INC $31        ja, Zeigeradresse auf nächsten Puffer
8F02  4C E1 8E  JMP $8EE1      weiter vergleichen
8F05[2] A9 D0   LDA #$D0       %11010000 'Forced Interrupt'
8F07  8D 00 20  STA $2000      an Kontroller; Vergleichen beenden
8F0A  20 83 A4  JSR $A483      ca. 80 Taktzyklen Verzögerung
8F0D  A2 07     LDX #$07       Fehlernummer für 'verify error'
8F0F  2C        .byte $2C      nächste 2 Bytes überspringen (Bitbefehl)
8F10[1] A2 00   LDX #$00       Fehlernummer für 'Ok'
8F12  8E B0 01  STX $01B0      Nummer merken
8F15  4C 61 88  JMP $8861      warten bis Kommando zu Ende
------------------------------------------------------------------------
[8900/Vektor: 86E0]
CP/M-Sektor auf Leerinhalt prüfen
8F18  A9 03     LDA #$03       aktuellen Pufferzeiger $30/$31
8F1A  85 31     STA $31        auf Startadresse von
8F1C  A0 00     LDY #$00       Puffer 0 ($0300)
8F1E  84 30     STY $30        setzen
8F20  A6 44     LDX $44        Zahl der Abschnitte pro Sektor
8F22  AC 71 02  LDY $0271      Länge eines Abschnitts
8F25  A9 88     LDA #$88       %10001000 'Read Sector' (Sektor lesen)
8F27  20 4E 88  JSR $884E      Kommando an CP/M-Kontroller übergeben
8F2A[3] AD 00 20 LDA $2000     Statusregister holen
8F2D  29 03     AND #$03       und Kommandobits isolieren
8F2F  4A        LSR A          Bit für 'Busy' prüfen
8F30  90 1A     BCC $8F4C      wird Kommando noch ausgeführt ?
8F32  29 01     AND #$01       ja, Flag 'für Daten bereit' prüfen
8F34  F0 F4     BEQ $8F2A      warten bis Datenbyte fertig ist ?
8F36  AD 03 20  LDA $2003      Byte von Diskette lesen
8F39  CD 0A 02  CMP $020A      und mit Wert für Leerbyte vergleichen
8F3C  D0 0E     BNE $8F4C      identisch ?
8F3E  88        DEY            ja, nächstes Byte
8F3F  10 E9     BPL $8F2A      schon ganzer Abschnitt verglichen ?
```

```
8F41      CA          DEX             ja, Zahl der Sektorabschnitte
8F42      F0 13       BEQ $8F57       noch ein Abschnitt ?
8F44      AC 71 02    LDY $0271       ja, Zähler wieder setzen
8F47      E6 31       INC $31         Pufferzeiger auf nächsten Puffer
8F49      4C 2A 8F    JMP $8F2A       weiter testen
8F4C[2]   A9 D0       LDA #$D0        '11010000' 'Forced Interrupt' an
8F4E      8D 00 20    STA $2000       Kontroller; Kommando abbrechen
8F51      20 83 A4    JSR $A483       ca. 80 Taktzyklen warten
8F54      A2 07       LDX #$07        Fehlernummer für 'verify error'
8F56      2C          .byte $2C       nächste 2 Bytes überspringen
8F57[1]   A2 00       LDX #$02        Fehlernummer für 'Header nicht gefunden'
8F59      8E B0 01    STX $01B0       Nummer setzen
8F5C      4C 61 88    JMP $8861       warten bis Kommando beendet
-----------------------------------------------------------------------
[Vektor: 86E2]
alle CP/M Header lesen und Sektorfolge feststellen
8F5F      08          PHP             Prozessorstatus retten
8F60      78          SEI             Diskcontrolleraufruf abschalten
8F61      20 EF 89    JSR $89EF       Kopf auf Spur0 setzen
8F64      24 3B       BIT $3B         Flag für 'Spur setzen' prüfen
8F66      10 08       BPL $8F70       soll neue Spur angesteuert werden ?
8F68      AD 03 02    LDA $0203       ja, Spurnummer aus Befehlsstring holen
8F6B      85 67       STA $67         und als Zielsprur setzen
8F6D      20 BA 87    JSR $87BA       Kopf auf Spur positionieren
8F70[1]   A9 00       LDA #$00        Zähler für Sektorzahl
8F72      85 97       STA $97         löschen
8F74      20 27 8A    JSR $8A27       nächsten Header einlesen
8F77      AE B0 01    LDX $01B0       Rückmeldung holen und
8F7A      E0 02       CPX #$02        auf Fehlernummer prüfen
8F7C      B0 1F       BCS $8F9D       ist Lesevorgang fehlerfrei verlaufen ?
8F7E      A5 26       LDA $26         ja, Nummer des Sektors holen
8F80      85 96       STA $96         und als erste Sektornummer merken
8F82[1]   20 27 8A    JSR $8A27       nächsten Header einlesen
8F85      A5 26       LDA $26         Nummer des Sektors holen
8F87      A4 97       LDY $97         Zeiger auf aktuelle Sektorposition
8F89      99 0B 02    STA $020B,Y     Sektornummer in Tabelle eintragen
8F8C      E6 97       INC $97         Zeiger auf nächsten Sektoreintrag
8F8E      C0 1F       CPY #$1F        mit max. Sektorzahl (31) vergleichen
8F90      B0 0B       BCS $8F9D       ist Zahl der Sektoren erlaubt ?
8F92      C5 96       CMP $96         ja, auf erste Sektornummmer prüfen
8F94      D0 EC       BNE $8F82       ist der erste Sektor wieder erreicht ?
```

```
8F96  A5 24     LDA $24        ja, Nummer der Spur aus Header holen
8F98  85 67     STA $67        und als aktuelle Zielspur setzen
8F9A  A2 00     LDX #$00       Wert für 'Ok' Meldung
8F9C  2C        .byte $2C      nächste 2 Bytes überspringen (Bitbefehl)
8F9D  A2 02     LDX #$02       Fehlernummer für 'Header nicht gefunden'
8F9F  8E B0 01  STX $01B0      Rückmeldung setzen
8FA2  28        PLP            Prozessorstatus wieder herstellen
8FA3  60        RTS            zurück zur aufrufenden Routine
```

[8FF1]
'S'-Kommando (Sektor) : Sektorversatz für Commodore-Disketten setzen

```
8FA4  AD 04 02  LDA $0204      5. Zeichen aus Befehlsstring holen
8FA7  85 69     STA $69        und als neuen Sektorversatz setzen
8FA9  60        RTS            zurück zur aufrufenden Routine
```

[8FF5]
'R'-Kommando (Read) : Zahl der Leseversuche festlegen

```
8FAA  AD 04 02  LDA $0204      5. Zeichen aus Befehlsstring holen und
8FAD  85 6A     STA $6A        als neue Zahl der Leseversuche setzen
8FAF  60        RTS            zurück zur aufrufenden Routine
```

[8FF9]
'T'-Kommando (Test) : Prüfsumme des ROM testen

```
8FB0  4C 4E 92  JMP $924E      Prüfsumme berechnen
```

[9001]
'H'-Kommando (Head) : Kopf auf angegebene Diskettenseite setzen
(nur im 1541 Modus)

```
8FB3  78        SEI            Bus- und Kontrolleraufruf abschalten
8FB4  AD 0F 18  LDA $180F      Steuerregister holen
8FB7  29 20     AND #$20       und Flag für Betriebsmodus holen
8FB9  D0 66     BNE $9021      ist Laufwerk im 1541 Modus ?
8FBB  AD 04 02  LDA $0204      ja, 5. Zeichen aus Befehlsstring holen
8FBE  C9 31     CMP #$31       und mit '1' vergleichen
8FC0  F0 12     BEQ $8FD4      soll Kopf auf Seite 2 gesetzt werden ?
8FC2  C9 30     CMP #$30       nein, mit '0' vergleichen
8FC4  D0 5B     BNE $9021      soll Kopf auf Seite 1 gesetzt werden ?
8FC6  AD 0F 18  LDA $180F      ja, Steuerregister holen
8FC9  29 FB     AND #$FB       und Kopfelektronik auf Seite 1
8FCB  8D 0F 18  STA $180F      schalten
8FCE  58        CLI            Bus- und Kontrolleraufruf einschalten
```

```
8FCF  24 3B      BIT $3B         Flags in Befehlsnummer testen
8FD1  10 0E      BPL $8FE1       soll Diskette initialisiert werden ?
8FD3  60         RTS             nein, zurück zur aufrufenden Routine
--------------------------------------------------------------------------

[8FC0]
Kopf auf Seite 2 anwählen
8FD4  AD 0F 18   LDA $180F       Steuerregister holen und
8FD7  09 04      ORA #$04        Bit für Kopfelektronik auf
8FD9  8D 0F 18   STA $180F       Seite 2 stellen
8FDC  58         CLI             Bus- und Kontrolleraufruf einschalten
8FDD  24 3B      BIT $3B         Flags in Befehlsnummer testen
8FDF  30 03      BMI $8FE4       soll Diskette initialisiert werden ?
8FE1  4C 42 D0   JMP $D042       ja, BAM von Diskette einlesen
8FE4  60         RTS             zurück zur aufrufenden Routine
--------------------------------------------------------------------------

[Einsprung über Vektor 80C2 durch Routine 8030]
Zusatz-Steuerfunktionen dekodieren
8FE5  AE 74 02   LDX $0274       Länge des Befehlsstrings feststellen
8FE8  E0 04      CPX #$04        und prüfen ob 4 Zeichen angeben sind
8FEA  90 35      BCC $9021       Befehl mindestens 4 Zeichen lang ?
8FEC  AD 03 02   LDA $0203       ja, 4. Zeichen aus Befehl holen
8FEF  C9 53      CMP #$53        und mit 'S' vergleichen
8FF1  F0 B1      BEQ $8FA4       soll Sektorversatz gesetzt werden ?
8FF3  C9 52      CMP #$52        nein, mit 'R' vergleichen
8FF5  F0 B3      BEQ $8FAA       Zahl der Leseversuche setzen ?
8FF7  C9 54      CMP #$54        nein, mit 'T' vergleichen
8FF9  F0 B5      BEQ $8FB0       ROM-Prüfsumme testen ?
8FFB  C9 4D      CMP #$4D        nein, mit 'M' vergleichen
8FFD  F0 27      BEQ $9026       1541/1571 Betrieb umschalten ?
8FFF  C9 48      CMP #$48        nein, mit 'H' vergleichen
9001  F0 B0      BEQ $8FB3       soll Diskettenseite angewählt werden ?
--------------------------------------------------------------------------

Geräteadresse setzen (Nummer in A)
9003  A8         TAY             Geräteadresse merken
9004  C0 04      CPY #$04        mit minimaler IEC-Adresse vergleichen
9006  90 19      BCC $9021       ist neue Adresse kleiner ?
9008  C0 1F      CPY #$1F        nein, auf maximale IEC-Adresse prüfen
900A  B0 15      BCS $9021       ist neue Adresse im erlaubten Bereich ?
900C  A9 40      LDA #$40        ja, Kennzeichen für Talk
900E  85 78      STA $78         setzen
9010  A9 20      LDA #$20        Kennzeichen für Listen
```

```
9012  85 77     STA $77       setzen
9014  98        TYA           neue Geräteadresse holen
9015  18        CLC           und damit neue Adresse für
9016  65 78     ADC $78       Talkaufruf erstellen
9018  85 78     STA $78       Adresse setzen
901A  98        TYA           neue Geräteadresse holen
901B  18        CLC           und damit neue Adresse für
901C  65 77     ADC $77       Listenaufruf erstellen
901E  85 77     STA $77       Adresse setzen
9020  60        RTS           zurück zur aufrufenden Routine
-----------------------------------------------------------------
[8FB9/8FC4/8FEA/9006/900A/9030]
9021  A9 31     LDA #$31      Fehlermeldung
9023  4C C8 C1  JMP $C1C8     '31 Syntax Error' ausgeben
-----------------------------------------------------------------
[8FFD]
'M'-Kommando (Mode) : 1541 / 1571 Betriebsmodus umschalten
9026  78        SEI           Bus- und Kontrolleraufruf abschalten
9027  AD 04 02  LDA $0204     5. Zeichen aus Befehlsstring holen
902A  C9 31     CMP #$31      und mit '1' vergleichen
902C  F0 20     BEQ $904E     in 1571 Modus schalten ?
902E  C9 30     CMP #$30      nein, mit '0' vergleichen
9030  D0 EF     BNE $9021     in 1541 Modus schalten ?
-----------------------------------------------------------------
auf 1541 Modus schalten
9032  AD 0F 18  LDA $180F     ja, Steuerregister holen
9035  29 DF     AND #$DF      Steuer- und Buselektronik
9037  8D 0F 18  STA $180F     auf 1541 umschalten (1 MHz Takt)
903A  20 83 A4  JSR $A483     80 Taktzyklen Verzögerung
903D  20 82 FF  JSR $FF82     1541 Modus initialisieren
9040  AD AF 02  LDA $02AF     Flag für
9043  09 80     ORA #$80      '1541 IRQ Routine' ($9D88)
9045  8D AF 02  STA $02AF     setzen
9048  58        CLI           Bus- und Kontrolleraufruf einschalten
9049  24 3B     BIT $3B       Flags in Befehlsnummer testen
904B  10 2F     BPL $907C     soll Diskette initialisiert werden ?
904D  60        RTS           nein, zurück zur aufrufenden Routine
-----------------------------------------------------------------
```

```
[902C]
auf 1571 Modus umschalten
904E  AD 0F 18   LDA $180F       Steuerregister holen und
9051  09 20      ORA #$20        Bus- und Betriebselektronik auf
9053  8D 0F 18   STA $180F       1571 schalten (2 MHz Takt)
9056  20 83 A4   JSR $A483       80 Taktzyklen Verzögerung
9059  A9 DE      LDA #$DE        IRQ Vektor in $02A9/$02AA zum
905B  8D A9 02   STA $02A9       Jobschleifenaufruf
905E  A9 9D      LDA #$9D        auf 1571 Routine in $9DDE
9060  8D AA 02   STA $02AA       richten
9063  A9 40      LDA #$40        Timer 1 (High-Byte)
9065  8D 07 1C   STA $1C07       auf etwa 8 ms
9068  8D 05 1C   STA $1C05       setzen
906B  AD AF 02   LDA $02AF       Flag für
906E  29 7F      AND #$7F        'IRQ von 1541 auf 1571 umschalten'
9070  8D AF 02   STA $02AF       setzen
9073  A9 00      LDA #$00        Flag für aktuellen Kopfmodus
9075  85 62      STA $62         löschen
9077  58         CLI             Bus- und Kontrolleraufruf einschalten
9078  24 3B      BIT $3B         Befehlsnummer testen
907A  30 03      BMI $907F       soll Diskette initialisiert werden ?
907C  4C 42 D0   JMP $D042       ja, BAM von Diskette lesen
907F  60         RTS             zurück zur aufrufeneden Routine
-----------------------------------------------------------------------------
[BF66/Einsprung über Vektor in 80CC durch Routine 8030]
Datei über 1571 Bus Schnell-Laden (PRG, SEQ oder USR)
9080  20 CE 81   JSR $81CE       1571 Bus auf Ausgang schalten
9083  20 EA 91   JSR $91EA       Dateiname vorbereiten
9086  B0 5F      BCS $90E7       Name fehlerhaft ?
9088  20 3D C6   JSR $C63D       nein, Diskette initialisieren
908B  A5 FF      LDA $FF         Flag für Laufwerksstatus holen
908D  D0 58      BNE $90E7       ist Laufwerk bereit ?
908F  A5 37      LDA $37         ja, Busstatus holen und Flags
9091  09 81      ORA #$81        für '1571 Modus' und 'letzter Sektor'
9093  85 37      STA $37         setzen
9095  20 CA 91   JSR $91CA       Kanal- und Pufferparameter setzen
9098  AD 00 02   LDA $0200       erstes Zeichen des Dateinamens holen
909B  C9 2A      CMP #$2A        und mit Joker '*' vergleichen
909D  D0 0F      BNE $90AE       zuletzt geladene Datei laden ?
909F  A5 7E      LDA $7E         ja, Nummer der letzten Spur holen
90A1  F0 0B      BEQ $90AE       ist Spurnummer angegeben ?
```

```
90A3  48        PHA          ja, Nummer merken
90A4  AD 6F 02  LDA $026F    Nummer des letzen Sektors
90A7  8D 85 02  STA $0285    holen und in Tabelle eintragen
90AA  68        PLA          letzte Spurnummer holen
90AB  4C EC 90  JMP $90EC    Datei laden
90AE  A9 00     LDA #$00     Zeiger und Register löschen :
90B0  A8        TAY          [Fehler siehe 7.1.5]
90B1  AA        TAX          [unnötige Initialisierung]
90B2  8D 8E 02  STA $028E    Nummer des letzten Laufwerks
90B5  8D 7A 02  STA $027A    Zeiger auf ersten Dateinamen
90B8  20 12 C3  JSR $C312    Laufwerksnummer aus Befehlsstring holen
90BB  AD 78 02  LDA $0278    Zahl der gefundenen Dateinamen
90BE  48        PHA          retten
90BF  A9 01     LDA #$01     und nur einen Dateinamen
90C1  8D 78 02  STA $0278    erlauben
90C4  A9 FF     LDA #$FF     Zeiger in Directorypuffer
90C6  85 86     STA $86      löschen
90C8  20 4F C4  JSR $C44F    Eintrag im Directory suchen
90CB  68        PLA          Zeiger mit Zahl der
90CC  8D 78 02  STA $0278    Dateinamen wieder holen
90CF  A5 37     LDA $37      Busstatus holen und
90D1  29 7F     AND #$7F     Flag für '1571 Modus'
90D3  85 37     STA $37      löschen
90D5  24 3B     BIT $3B      Befehlsnummer holen und Flag testen
90D7  30 06     BMI $90DF    soll auf 'PRG' Datei geprüft werden ?
90D9  A5 E7     LDA $E7      Dateityp des Dateieintrags feststellen
90DB  C9 02     CMP #$02     und mit Kennzeichen für PRG vergleichen
90DD  D0 05     BNE $90E4    ist Eintrag eine PRG-Datei ?
90DF  AD 80 02  LDA $0280    ja, Spurnummer des ersten Dateisektors
90E2  D0 08     BNE $90EC    wurde Eintrag im Directory gefunden ?
90E4  A2 02     LDX #$02     Fehlernummer für 'File Not Found'
90E6  2C        .byte $2C    nächste 2 Bytes überspringen (Bitbefehl)
90E7  A2 0F     LDX #$0F     Fehlernummer für 'Drive Not Ready'
90E9  4C AD 91  JMP $91AD    Fehler auf 1571 Bus ausgeben
90EC  85 7E     STA $7E      letzte Spur merken
90EE  48        PHA          und Nummer retten
90EF  20 DA 91  JSR $91DA    Zeiger auf Jobtabelle berechnen
90F2  68        PLA          letzte Spurnummer wieder holen
90F3  AE B0 02  LDX $02B0    Zeiger auf Jobtabelle holen
90F6  95 06     STA $06,X    und Spur des Jobs festlegen
90F8  AD 85 02  LDA $0285    letzte Sektornummer holen
```

```
90FB  8D 6F 02   STA $026F      und merken
90FE  95 07      STA $07,X      Sektornummer an Jobschleife übergeben
9100  A9 80      LDA #$80       Jobcode für Sektor lesen
9102  8D 02 02   STA $0202      setzen
9105  85 5F      STA $5F        und als aktuellen Jobcode merken
9107  58         CLI            Bus- und Kontrolleraufruf einschalten
9108  A6 F9      LDX $F9        Nummer des aktuellen Puffers holen
910A  A5 5F      LDA $5F        und aktuellen Jobcode
910C  95 00      STA $00,X      an Jobschleife übergeben
910E  20 4B 86   JSR $864B      Job ausführen
9111  E0 02      CPX #$02       Rückmeldung auf 'Ok' prüfen
9113  90 03      BCC $9118      ist Job fehlerfrei verlaufen ?
9115  4C 99 91   JMP $9199      nein, Rückmeldung ausgeben
9118  78         SEI            Bus- und Kontrolleraufruf abschalten
9119  A0 00      LDY #$00       Pufferzeiger auf erstes Byte im Sektor
911B  B1 94      LDA ($94),Y    Byte aus Puffer holen
911D  F0 2F      BEQ $914E      ist dies der letzte Sektor ?
911F  A5 37      LDA $37        nein, Busstatus holen
9121  29 FE      AND #$FE       und Flag für 'letzten Sektor'
9123  85 37      STA $37        löschen
9125  20 28 92   JSR $9228      letzte 'Ok' Meldung auf 1571 Bus geben
9128  A0 02      LDY #$02       Pufferzeiger auf erstes Datenbyte
912A  B1 94      LDA ($94),Y    Byte aus Puffer holen
912C  AA         TAX            und für Ausgabe vorbereiten
912D  20 28 92   JSR $9228      Byte auf 1571 Bus ausgeben
9130  C8         INY            Pufferzeiger auf nächstes Byte richten
9131  D0 F7      BNE $912A      schon ganzer Puffer übertragen ?
9133  AE B0 02   LDX $02B0      ja, Zeiger in Jobtabelle holen
9136  B1 94      LDA ($94),Y    Spur des nächsten Sektors aus Puffer
9138  D5 06      CMP $06,X      mit Spur des letzten Jobs vergleichen
913A  F0 03      BEQ $913F      nächster Sektor auf gleicher Spur ?
913C  A0 80      LDY #$80       nein, Jobcode für 'Sektor lesen'
913E  2C         .byte $2C      nächste 2 Bytes überspringen (Bitbefehl)
913F  A0 88      LDY #$88       Jobcode 'Sektor auf gleicher Spur lesen'
9141  84 5F      STY $5F        Jobcode setzen
9143  95 06      STA $06,X      und Spurnummer an Jobschleife übergeben
9145  A0 01      LDY #$01       Zeiger auf Nummer des nächsten Sektors
9147  B1 94      LDA ($94),Y    Byte aus Verkettungsbytes holen
9149  95 07      STA $07,X      und an Jobschleife übergeben
914B  4C 07 91   JMP $9107      nächsten Sektor übertragen
914E  A2 1F      LDX #$1F       Rückmeldung für 'letzter Sektor'
```

```
9150   20 28 92   JSR $9228     auf 1571 Bus ausgeben
9153   A9 01      LDA #$01      Flag für 'nur ein Sektor'
9155   24 37      BIT $37       in Busstatusbyte testen
9157   F0 1E      BEQ $9177     hat das Programm nur einen Block ?
9159   A8         TAY           ja, Pufferzeiger setzen
915A   B1 94      LDA ($94),Y   Zahl der gültigen Datenbytes des Sektors
915C   38         SEC           holen und Bytes
915D   E9 03      SBC #$03      für Startaddresse und Verkettungs-
915F   85 46      STA $46       bytes abziehen
9161   AA         TAX           Zahl der noch zu übertragenden Bytes
9162   20 28 92   JSR $9228     auf 1571 Bus übergeben
9165   C8         INY           Pufferzeiger auf Programmstartadresse
9166   B1 94      LDA ($94),Y   Low-Byte der Startadresse holen
9168   AA         TAX           und als auszugebendes Zeichen setzen
9169   20 28 92   JSR $9228     Byte auf 1571 Bus ausgeben
916C   C8         INY           Pufferzeiger auf High-Byte richten
916D   B1 94      LDA ($94),Y   und Byte aus Puffer holen
916F   AA         TAX           Rest der Startadresse auf
9170   20 28 92   JSR $9228     1571 Bus ausgeben
9173   A0 04      LDY #$04      Pufferzeiger auf Beginn der Daten setzen
9175   D0 0D      BNE $9184     immer Sprung nahc $9184
9177[1] A0 01     LDY #$01      Zeiger auf noch gültige Datenbytes
9179   B1 94      LDA ($94),Y   Zahl der Datenbytes aus Puffer holen
917B   AA         TAX           und merken
917C   CA         DEX           Zahl der noch
917D   86 46      STX $46       folgenden Datenbytes
917F   20 28 92   JSR $9228     auf 1571 Bus ausgeben
9182   A0 02      LDY #$02      Zeiger auf Start des Datenbereichs
9184[2] B1 94     LDA ($94),Y   Byte aus Puffer holen
9186   AA         TAX           und für Ausgabe vorbereiten
9187   20 28 92   JSR $9228     Byte auf 1571 Bus ausgeben
918A   C8         INY           Pufferzeiger auf nächstes Byte richten
918B   C6 46      DEC $46       Zahl der noch zu übertragenden Bytes
918D   D0 F5      BNE $9184     schon alle übertragen ?
918F   A9 00      LDA #$00      ja, Sekundäradresse für Load
9191   85 83      STA $83       setzen
9193   20 C0 DA   JSR $DAC0     Datei schließen
9196   4C 94 C1   JMP $C194     Rückmeldung bereitstellen
----------------------------------------------------------------------
```

```
[9115/A9CF]
Fehlermeldung ausgeben
9199  78        SEI          Bus- und Kontrolleraufruf abschalten
919A  86 46     STX $46      Fehlernummer auf
919C  20 28 92  JSR $9228    1571 Bus ausgeben
919F  A9 00     LDA #$00     Sekundäradresse für Load
91A1  85 83     STA $83      setzen
91A3  20 C0 DA  JSR $DAC0    Datei schließen
91A6  A6 F9     LDX $F9      Nummer des aktuellen Puffers
91A8  A5 46     LDA $46      Nummer des Fehlers
91AA  4C 0A E6  JMP $E60A    Klartextfehlermeldung bereitstellen
------------------------------------------------------------------
[90E9]
Fehler bei Load ausgeben
91AD  78        SEI          Bus- und Kontrolleraufruf abschalten
91AE  86 46     STX $46      Fehlernummer merken
91B0  A2 02     LDX #$02     Fehlernummer für 'File Not Found'
91B2  20 28 92  JSR $9228    auf 1571 Bus ausgeben
91B5  A9 00     LDA #$00     Sekundäraddresse für Load
91B7  85 83     STA $83      setzen
91B9  20 C0 DA  JSR $DAC0    Datei schließen
91BC  A5 46     LDA $46      Fehlernummer wieder holen
91BE  C9 02     CMP #$02     und auf 'File Not Found' prüfen
91C0  F0 03     BEQ $91C5    identisch ?
91C2  A9 74     LDA #$74     nein, Nummer für 'Drive Not Ready'
91C4  2C A9 62  .byte $2C    nächste 2 Bytes überspringen (Bitbefehl)
91C5  A9 62     LDA #$62     Nummer für 'File Not Found'
91C7  4C C8 C1  JMP $C1C8    Klartextmeldung bereitstellen
------------------------------------------------------------------
[9095]
Kanal und Puffer für Fast Load belegen
91CA  A9 00     LDA #$00     Sekundäradresse für Load
91CC  85 83     STA $83      setzen
91CE  A9 01     LDA #$01     Zahl der zu belegenden Puffer
91D0  20 E2 D1  JSR $D1E2    Puffer und Kanal belegen
91D3  AA        TAX          Nummer des zugeordneten Puffers holen
91D4  BD E0 FE  LDA $FEE0,X  und High-Byte der Pufferadresse
91D7  85 95     STA $95      in Pufferzeiger übernehmen
91D9  60        RTS          zurück zur aufrufenden Routine
------------------------------------------------------------------
```

[90EF]

Zeiger auf Jobtabelle der Spur- und Sektornummern ermitteln

```
91DA  A5 95     LDA $95        High-Byte des Pufferzeigers
91DC  38        SEC            holen und aus physikalischer Adresse
91DD  E9 03     SBC #$03       logische Puffernummer errechnen
91DF  85 F9     STA $F9        und als aktuelle Puffernummer setzen
91E1  0A        ASL A          Nummer verdoppeln (da 2-Byte Tabellle)
91E2  8D B0 02  STA $02B0      und merken
91E5  A9 00     LDA #$00       Low-Byte des Pufferzeigers
91E7  85 94     STA $94        auf Pufferstart zurücksetzen
91E9  60        RTS            zurück zur aufrufenden Routine
```

[9083]

Dateinamen an Anfang des Eingabepuffers verschieben

```
91EA  A0 03     LDY #$03       Zeiger auf Beginn des Dateinamens
91EC  AD 74 02  LDA $0274      Länge des Befehlsstring holen
91EF  38        SEC            und Zeichen für
91F0  E9 03     SBC #$03       'U0'-Befehl abziehen
91F2  8D 74 02  STA $0274      Länge des Dateinamens merken
91F5  AD 04 02  LDA $0204      zweites Zeichen des Namens
91F8  C9 3A     CMP #$3A       auf Doppelpunkt ':' prüfen
91FA  D0 0E     BNE $920A      Laufwerksnennung vorhanden ?
91FC  AD 03 02  LDA $0203      ja, Nummer des Laufwerks holen
91FF  AA        TAX            und merken
9200  29 30     AND #$30       Nummer auf ASCII-Ziffer
9202  C9 30     CMP #$30       prüfen
9204  D0 04     BNE $920A      liegt eine Ziffer vor ?
9206  E0 31     CPX #$31       ja, mit '1' vergleichen
9208  F0 1C     BEQ $9226      Laufwerk 1 angewählt ?
920A  AD 03 02  LDA $0203      ja, mit '0' vergleichen
920D  C9 3A     CMP #$3A       mit ':' vergleichen
920F  D0 04     BNE $9215      auch ein Doppelpunkt ?
9211  CE 74 02  DEC $0274      ja, Länge des Dateinamens verkürzen
9214  C8        INY            Zeiger auf nächstes Pufferbyte
9215  A2 00     LDX #$00       Zeiger auf Beginn des Eingabepuffers
9217  B9 00 02  LDA $0200,Y    Dateinamen
921A  9D 00 02  STA $0200,X    an Beginn des Puffers verschieben
921D  C8        INY            Pufferzeiger
921E  E8        INX            auf nächstes Zeichen richten
921F  EC 74 02  CPX $0274      mit Ende des Dateinamens vergleichen
9222  D0 F3     BNE $9217      schon ganzer Name verschoben
```

```
9224    18         CLC           ja, Flag für Name fehlerfrei
9225    24         .byte $24     nächstes Byte überspringen (Bit-Befehl)
9226 1  38         SEC           Flag für fehlerhafte Laufwerksnennung
9227    60         RTS           zurück zur aufrufenden Routine
------------------------------------------------------------------------
[9125/912D/9150/9162/9169/9170/917F/9187/919C/91B2/A9EA]
Byte bei Fast Load auf 1571 Bus geben
9228    AD 00 18   LDA $1800     Bussteuerregister holen
922B    CD 00 18   CMP $1800     und auf konstanten Zustand warten
922E    D0 F8      BNE $9228     keine Änderung ?
9230    29 FF      AND #$FF      ja, Prozessorflags (N/Z) setzen
9232    30 17      BMI $924B     ist ATN Eingang gesetzt ?
9234    45 37      EOR $37       nein, Busstatusflag holen
9236    29 04      AND #$04      und auf erwarteten Clockzustand prüfen
9238    F0 EE      BEQ $9228     hat Clock seit letztem mal gewechselt ?
923A    8E 0C 40   STX $400C     ja, Byte in Ausgaberegister schreiben
923D    A5 37      LDA $37       Busstatus holen und
923F    49 04      EOR #$04      Flag für 'Zustand des Clock-Eingangs'
9241    85 37      STA $37       auf nächsten Wert setzen
9243    A9 08      LDA #$08      Flag für 'Ausgaberegister leer'
9245 1  2C 0D 40   BIT $400D     prüfen
9248    F0 FB      BEQ $9245     ist Byte übertragen ?
924A    60         RTS           ja, zurück zur aufrufenden Routine
924B 1  4C B3 A7   JMP $A7B3     ATN-Kommando bearbeiten
------------------------------------------------------------------------
[8FB0/BF69]
Prüfsumme des ROM berechnen und ROM prüfen
924E    08         PHP           Prozessorstatus retten
924F    78         SEI           Bus- und Kontrolleraufruf abschalten
9250    A2 00      LDX #$00      Ergebnisregister
9252    86 00      STX $00       für errechnete Prüfsumme
9254    86 01      STX $01       löschen
9256    A9 03      LDA #$03      Startadresse des ROM (Low-Byte)
9258    85 75      STA $75       setzen
925A    A8         TAY           Zeiger setzen
925B    A9 80      LDA #$80      High-Byte der ROM-Adresse
925D    85 76      STA $76       festlegen
925F 2  B1 75      LDA ($75),Y   Byte aus ROM holen
9261    85 02      STA $02       und merken
9263    A2 08      LDX #$08      Zahl der Bits pro Byte
9265 1  A5 02      LDA $02       ROM-Byte holen und ein Bit
```

```
9267  29 01     AND #$01     isolieren
9269  85 03     STA $03      Bit in Zwischenspeicher übernehmen
926B  A5 01     LDA $01      Bit15 des
926D  10 02     BPL $9271    Prüfsummenregisters
926F  E6 03     INC $03      dazuaddieren
9271  6A        ROR A        Bit11 des
9272  90 02     BCC $9276    Prüfsummenregisters
9274  E6 03     INC $03      dazuaddieren
9276  6A        ROR A        Bit8 des
9277  6A        ROR A        16-Bit Prüfsummenregisters
9278  6A        ROR A        in $00 und $01
9279  90 02     BCC $927D    zum Zwischenspeicher
927B  E6 03     INC $03      dazurechnen
927D  A5 00     LDA $00      Bit6 des
927F  2A        ROL A        Prüfsummenregisters
9280  2A        ROL A        holen und
9281  90 02     BCC $9285    zum Zwischenspeicher
9283  E6 03     INC $03      addieren
9285  66 03     ROR $03      Prüfsummenregister um ein Bit nach
9287  26 00     ROL $00      links verschieben und in freie Stelle
9289  26 01     ROL $01      Bit0 des Zwischenspeichers  übertragen
928B  66 02     ROR $02      nächstes Bit des ROM-Bytes anwählen
928D  CA        DEX          Zahl der Bits pro Byte
928E  D0 D5     BNE $9265    schon ganzes Byte verarbeitet ?
9290  E6 75     INC $75      ja, Zeiger auf aktuelles
9292  D0 CB     BNE $925F    Byte im ROM
9294  E6 76     INC $76      auf nächstes Position richten
9296  D0 C7     BNE $925F    ist Endadresse $FFFF erreicht ?
9298  88        DEY          ja, Zeiger
9299  88        DEY          auf Null
929A  88        DEY          zurücksetzen
929B  A5 00     LDA $00      erstes errechnetes Byte auf
929D  CD 00 80  CMP $8000    richtige Prüfsumme testen
92A0  D0 11     BNE $92B3    liegt ein Fehler vor ?
92A2  A5 01     LDA $01      nein, zweites errechnetes Byte auf
92A4  CD 01 80  CMP $8001    richtige Prüfsumme testen
92A7  D0 0A     BNE $92B3    ist ein Prüfsummenfehler aufgetreten ?
92A9  84 00     STY $00      nein, Prüfsummenregister
92AB  84 01     STY $01      und die diversen
92AD  84 02     STY $02      Zwischenspeicher wieder
92AF  84 03     STY $03      löschen
```

```
92B1   28         PLP          Prozessorstatus wieder herstellen
92B2   60         RTS          zurück zur aufrufenden Routine
92B3   A2 03      LDX #$03     Flag für Hardwarefehler
92B5   86 6F      STX $6F      initialisieren
92B7   4C 71 EA   JMP $EA71    Hardwarefehler melden (LED-Blinken)
--------------------------------------------------------------------
[9E08/9E11/BF09]
1571 Jobschleife
92BA   BA         TSX          aktuellen Stapelzeiger
92BB   86 49      STX $49      merken
92BD   2C 04 1C   BIT $1C04    Timer zurücksetzen
92C0   AD 0C 1C   LDA $1C0C    CA2-Ausgang 'SOE'
92C3   09 0E      ORA #$0E     (Seriell Output Enable)
92C5   8D 0C 1C   STA $1C0C    auf High setzen
92C8   A0 05      LDY #$05     Zahl der Puffer
92CA   B9 00 00   LDA $0000,Y  Jobcode des Puffers holen
92CD   30 06      BMI $92D5    ist Jobcode vorhanden ?
92CF   88         DEY          nein, nächsten Puffer prüfen
92D0   10 F8      BPL $92CA    schon alle Puffer geprüft ?
92D2   4C CA 99   JMP $99CA    ja, Stepperkommandos ausführen
92D5   C9 88      CMP #$88     Jobcode 'Sektor auf gleicher Spur lesen'
92D7   D0 03      BNE $92DC    identisch ?
92D9   4C 0D 96   JMP $960D    ja, Sektor in Puffer einlesen
92DC   C9 D0      CMP #$D0     Jobcode für 'Programm ausführen'
92DE   D0 03      BNE $92E3    identisch ?
92E0   4C A2 93   JMP $93A2    ja, Programm im Puffer starten
92E3   29 01      AND #$01     Nummer des gewünschten Laufwerks holen
92E5   F0 07      BEQ $92EE    ist Laufwerk 0 angewählt ?
92E7   84 3F      STY $3F      nein, Nummer des Puffers merken
92E9   A9 0F      LDA #$0F     Fehlermeldung
92EB   4C B5 99   JMP $99B5    'Drive not Ready' zurückmelden
92EE   AA         TAX          Laufwerksnummer merken
92EF   C5 3E      CMP $3E      auf Nummer des aktiven Laufwerks prüfen
92F1   F0 08      BEQ $92FB    identisch ?
92F3   85 3E      STA $3E      nein, dann aktuelles Laufwerk neu setzen
92F5   20 7E F9   JSR $F97E    Laufwerksmotor einschalten
92F8   4C CA 99   JMP $99CA    Stepperkommandos ausführen
92FB   A5 20      LDA $20      Laufwerksstatus holen
92FD   30 03      BMI $9302    ist Laufwerk bereit ?
92FF   0A         ASL A        ja, Flag für Steppermotor testen
9300   10 03      BPL $9305    ist Kopf noch in Bewegung ?
```

```
9302[1] 4C CA 99   JMP $99CA      ja, Stepperfunktionen ausführen
9305[1] A9 20      LDA #$20       Flag für 'Motor an/Laufwerk bereit'
9307    85 20      STA $20        in Laufwerksstatus setzen
9309    A0 05      LDY #$05       Zahl der Puffer
930B    84 3F      STY $3F        aktuellen Puffer anwählen
930D[1] 20 D1 93   JSR $93D1      Pufferzeiger setzen und Jobcode holen
9310    30 1A      BMI $932C      ist ein Job vorhanden ?
9312[2] C6 3F      DEC $3F        nein, nächsten Puffer anwählen
9314    10 F7      BPL $930D      schon alle Puffer geprüft ?
9316    A4 41      LDY $41        Puffernummer des letzten Jobs holen
9318    20 D3 93   JSR $93D3      Pufferzeiger setzen
931B    A5 42      LDA $42        Nummer der anzusteuernden Spur
931D    85 4A      STA $4A        als Zielspur merken
931F    06 4A      ASL $4A        Zahl der Halbspurschritte errechnen
9321    A9 60      LDA #$60       Flag für 'Stepper an / Motor an'
9323    85 20      STA $20        in Laufwerksstatus setzen
9325    B1 32      LDA ($32),Y    Spur des Jobs holen
9327    85 22      STA $22        und merken
9329    4C CA 99   JMP $99CA      Spur ansteuern
932C[1] 29 01      AND #$01       Nummer des angewählten Laufwerks mit
932E    C5 3E      CMP $3E        aktueller Laufwerksnummer vergleichen
9330    D0 E0      BNE $9312      identisch ?
9332    A5 22      LDA $22        Nummer der aktuellen Spur prüfen
9334    F0 32      BEQ $9368      ist Zeiger gesetzt ?
9336    A5 22      LDA $22        ja, aktuelle Spur holen
9338    C9 24      CMP #$24       und mit maximaler Spur +1 vergleichen
(36)
933A    08         PHP            Ergebnis merken
933B    B1 32      LDA ($32),Y    Spur des Jobs mit maximaler Spur +1
933D    C9 24      CMP #$24       vergleichen
933F    6A         ROR A          Ergebnis in Bit7
9340    28         PLP            vorheriges Ergebnis im Carry
9341    29 80      AND #$80       letztes Prüfergebnis isolieren
9343    90 0B      BCC $9350      ist aktuelle Spur auf Seite 2 ?
9345    30 11      BMI $9358      ja, ist neue Spur auf Seite 1 ?
9347    A5 22      LDA $22        ja, Nummer der aktuellen Spur
9349    E9 23      SBC #$23       auf Seite 1 umrechnen
934B    85 22      STA $22        und merken
934D    4C 58 93   JMP $9358      Spurnummer weiter bearbeiten
9350[1] 10 06      BPL $9358      ist neue Spur auf Seite 2 ?
9352    A5 22      LDA $22        ja, aktuelle Spurnummer
```

```
9354     69 23      ADC #$23      auf Seite 2 umrechen und
9356     85 22      STA $22       merken
9358[3]  38         SEC           Differenz zwischen
9359     B1 32      LDA ($32),Y   neuer Spur und
935B     E5 22      SBC $22       aktueller Spur ausrechnen
935D     F0 09      BEQ $9368     ist Kopf schon auf gewünschter Spur ?
935F     85 42      STA $42       Zahl der zu fahrenden Schritte merken
9361     A5 3F      LDA $3F       Nummer des aktuellen Puffers holen
9363     85 41      STA $41       und merken
9365     4C 12 93   JMP $9312     nächsten Job bearbeiten
9368[2]  A2 04      LDX #$04      keine Funktion [Fehler siehe 7.1.5]
936A     B1 32      LDA ($32),Y   Nummer der Spur holen
936C     85 40      STA $40       und merken
936E     C9 24      CMP #$24      mit maximaler Spurnummer +1 vergleichen
9370     A8         TAY           und merken
9371     20 F3 93   JSR $93F3     entsprechende Seite anwählen
9374     98         TYA           Spur wieder holen
9375     90 02      BCC $9379     liegt Spur auf Seite 2
9377     E9 23      SBC #$23      ja, absolute Spur der Seite errechnen
9379[1]  AA         TAX           und merken
937A     BD 08 94   LDA $9408,X   Bitrate des Spurbereichs ermitteln
937D     85 43      STA $43       und setzen
937F     AD 00 1C   LDA $1C00     Laufwerkssteuerregister holen
9382     29 9F      AND #$9F      Bits für Aufzeichnungsrate
9384     05 43      ORA $43       neu setzen
9386     8D 00 1C   STA $1C00     und in Steuerregister setzen
9389     BD 2B 94   LDA $942B,X   Zahl der Sektoren der Spur feststellen
938C     85 43      STA $43       und speichern
938E     A5 45      LDA $45       Befehlbits des Jobcodes holen
9390     C9 40      CMP #$40      und auf 'Bump' prüfen
9392     F0 1C      BEQ $93B0     soll Kopf auf Spur0 gesetzt werden ?
9394     C9 60      CMP #$60      nein, auf 'Programm ausführen' testen
9396     F0 0A      BEQ $93A2     soll Pufferprogramm gestartet werden ?
9398     C9 70      CMP #$70      nein, auf 'Formatieren' prüfen
939A     F0 03      BEQ $939F     soll Diskette formatiert werden ?
939C     4C 4F 94   JMP $944F     nein, Sektorheader lesen
939F[1]  4C 29 9B   JMP $9B29     Diskette formatieren
--------------------------------------------------------------------------
```

```
[92E0/9396]  vgl. F36E
Programm im Puffer in Jobschleife einbinden
93A2  A5 3F      LDA $3F          Nummer des aktuellen Puffers
93A4  18         CLC              holen und
93A5  69 03      ADC #$03         physikalische Pufferadresse
93A7  85 31      STA $31          berechnen
93A9  A9 00      LDA #$00         Low-Byte auf
93AB  85 30      STA $30          Pufferanfang setzen
93AD  6C 30 00   JMP ($0030)      Programm ausführen
--------------------------------------------------------------------
Kopf auf Spur 0 zurücksetzen ('Bump')  [vgl. F37C]
93B0[1] A9 60    LDA #$60         Flag für 'Stepper an/Motor an'
93B2  85 20      STA $20          in Laufwerksstatus setzen
93B4  AD 00 1C   LDA $1C00        Steuerregister holen
93B7  29 FC      AND #$FC         und Steppersteuerbits
93B9  8D 00 1C   STA $1C00        löschen
93BC  A9 A4      LDA #$A4         Zahl der Spuren setzen (-36)
93BE  85 4A      STA $4A          die der Kopf zurückbewegt wird
93C0  AD B1 01   LDA $01B1        Flag für aktuelle Diskettenseite holen
93C3  30 03      BMI $93C8        ist Seite 1 angewählt ?
93C5  A9 01      LDA #$01         ja, erste Spurnummer setzen (1)
93C7  2C         .byte $2C        nächste 2 Bytes überspringen (Bitbefehl)
93C8[1] A9 24    LDA #$24         erste Spur der zweiten Seite (36)
93CA  85 22      STA $22          Spurnummer merken
93CC  A9 01      LDA #$01         Rückmeldung für 'Ok'
93CE  4C B5 99   JMP $99B5        übergeben
--------------------------------------------------------------------
[930D/94D3/9527/BF0F/93D3:9318]  vgl. F393
Pufferzeiger setzen und Jobcode des Puffers holen
93D1  A4 3F      LDY $3F          Nummer des aktuellen Puffers
93D3  B9 00 00   LDA $0000,Y      zugehörigen Jobcode holen
93D6  48         PHA              und merken
93D7  10 14      BPL $93ED        ist ein Befehl vorhanden ?
93D9  29 78      AND #$78         ja, Bit 3-6 isolieren und als
93DB  85 45      STA $45          signifikante Befehlsbits merken
93DD  98         TYA              Nummer des Puffers holen
93DE  0A         ASL A            und verdoppeln
93DF  69 06      ADC #$06         Zeiger auf Tabelle
93E1  85 32      STA $32          der Spur- und Sektorangaben
93E3  A9 00      LDA #$00         zu dem Job ($0006-$0011)
93E5  85 33      STA $33          setzen
```

```
93E7  98          TYA          Nummer des Puffers holen
93E8  18          CLC          und daraus die
93E9  69 03       ADC #$03     physikalische Speicheradresse
93EB  85 31       STA $31      berechnen
93ED  A0 00       LDY #$00     Adresse in Zeiger $30/$31
93EF  84 30       STY $30      ablegen
93F1  68          PLA          Jobcode wieder holen
93F2  60          RTS          zurück zur aufrufenden Routine
---------------------------------------------------------------------

[895C/9371/9B41]
Schreib-/Lesekopf auf der aktuellen Diskettenseite aktivieren
93F3  B0 03       BCS $93F8    ist Seite 2 angewählt ?
93F5  A9 00       LDA #$00     nein, Steuerbits für erste Seite
93F7  2C          .byte $2C    nächste 2 Bytes überspringen
93F8  A9 84       LDA #$84     Steuerbits für Seite 2 (%10000100)
93FA  8D B1 01    STA $01B1    Bits merken
93FD  AD 0F 18    LDA $180F    Steuerregister A holen
9400  29 FB       AND #$FB     und Bits neu
9402  0D B1 01    ORA $01B1    setzen
9405  8D 0F 18    STA $180F    Wert in Steuerregister schreiben
9408  60          RTS          zurück zur aufrufenden Routine
---------------------------------------------------------------------

[937A]  Steuerbits zur Aufzeichnungsrate auf jeder Spur

Bit6  Bit5  Spurbereich  Aufzeichnungsrate
 0     0      31 - 35     31250 Bytes/sec
 0     1      25 - 30     33333 Bytes/sec
 1     0      18 - 24     35714 Bytes/sec
 1     1       1 - 17     38461 Bytes/sec

9409 60 60 60 60 60 60 60 60 60 60 60 60 60 60 60 60
9419 60 40 40 40 40 40 40 40 20 20 20 20 20 20 00 00
9429 00 00 00
---------------------------------------------------------------------

[A82C/A8C2]  Zahl der Sektoren pro Spur im Commodore-Format
942C 15 15 15 15 15 15 15 15 15 15 15 15 15 15 15 15
943C 15 13 13 13 13 13 13 13 12 12 12 12 12 12 11 11
944C 11 11 11
---------------------------------------------------------------------
```

```
[939C/97F6]
Header eines Sektors suchen
944F  A9 5A     LDA #$5A     Zahl der Leseversuche (90)
9451  85 4B     STA $4B      festlegen
9453  20 54 97  JSR $9754    auf nächste Sync-Markierung warten
9456  2C 0F 18  BIT $180F    'Byte Ready'-Signal prüfen
9459  30 FB     BMI $9456    ist nächstes Byte bereit ?
945B  AD 01 1C  LDA $1C01    ja, GCR-Byte von Diskette lesen
945E  C9 52     CMP #$52     mit Kennzeichen für Header vergleichen
9460  D0 3E     BNE $94A0    ist das ein Sektorheader ?
9462  99 24 00  STA $0024,Y  ja, Byte in Headerpuffer
9465  C8        INY          Pufferzeiger auf nächstes Byte setzen
9466  2C 0F 18  BIT $180F    'Byte Ready'-Signal prüfen
9469  30 FB     BMI $9466    ist nächstes Byte bereit ?
946B  AD 01 1C  LDA $1C01    ja, GCR-Byte von Diskette lesen
946E  99 24 00  STA $0024,Y  Byte in Headerpuffer
9471  C8        INY          Pufferzeiger auf nächstes Byte setzen
9472  C0 08     CPY #$08     Zahl der Headerbytes
9474  D0 F0     BNE $9466    schon ganzen Header gelesen ?
9476  20 2F 95  JSR $952F    ja, Blockheader von GCR nach Binär
9479  A0 04     LDY #$04     Zahl der relevanten Headerbytes
947B  A9 00     LDA #$00     Prüfsumme der Bytes berechnen :
947D  59 16 00  EOR $0016,Y  Headerbyte einrechnen
9480  88        DEY          Zeiger auf nächstes Byte
9481  10 FA     BPL $947D    schon alle Bytes eingerechnet
9483  C9 00     CMP #$00     mit Wert für 'richtig' vergleichen
9485  D0 30     BNE $94B7    ist Prüfsumme fehlerfrei ?
9487  A5 18     LDA $18      ja, Spurnummer aus Header als
9489  85 22     STA $22      Nummer der aktuellen Spur setzen
948B  A5 45     LDA $45      Jobcodebefehlbits holen und auf
948D  C9 30     CMP #$30     Code für 'Sektor suchen' prüfen
948F  F0 18     BEQ $94A9    soll Sektorheader gesucht werden ?
9491  A5 12     LDA $12      nein, ID aus Sektorheader mit
9493  C5 16     CMP $16      aktueller ID vergleichen
9495  D0 1D     BNE $94B4    identisch ?
9497  A5 13     LDA $13      nächstes ID Zeichen
9499  C5 17     CMP $17      prüfen
949B  D0 17     BNE $94B4    trat ein Diskwechsel auf ?
949D  4C BC 94  JMP $94BC    nein, nächsten Job holen
94A0  C6 4B     DEC $4B      Zahl der Leseversuche
94A2  D0 AF     BNE $9453    noch einen Versuch durchführen ?
```

```
94A4    A9 02      LDA #$02       nein, Fehlernummer zu 'Header Not Found'
94A6    20 B5 99   JSR $99B5      Fehlermeldung ausgeben
94A9[1] A5 16      LDA $16        aktuelle ID aus Header
94AB    85 12      STA $12        übernehmen (erstes Zeichen)
94AD    A5 17      LDA $17        aktuelle ID aus Header
94AF    85 13      STA $13        übernehmen (zweites Zeichen)
94B1    A9 01      LDA #$01       Nummer für 'Ok' Meldung
94B3    2C         .byte $2C      nächste 2 Bytes überspringen (Bitbefehl)
94B4[2] A9 0B      LDA #$0B       Fehlernummer für 'id mismatch'
94B6    2C         .byte $2C      nächste 2 Bytes überspringen (Bitbefehl)
94B7[1] A9 09      LDA #$09       Fehlernummer für 'read error (27)'
94B9    4C B5 99   JMP $99B5      Rückmeldung übergeben
------------------------------------------------------------------------
```

[949D] vgl. F423
nächsten Job holen (Sektoroptimiert)
optimal ist ein Abstand von > 6 (lesen) oder 9-12 Sektoren (schreiben)

```
94BC    A9 7F      LDA #$7F       Zeiger für Differenz zum nächsten Job
94BE    85 4C      STA $4C        initialisieren
94C0    A5 19      LDA $19        Nummer des Sektors
94C2    18         CLC            aus Blockheader
94C3    69 02      ADC #$02       mit maximaler
94C5    C5 43      CMP $43        Sektornummer vergleichen
94C7    90 02      BCC $94CB      ist Nummer im erlaubten Bereich ?
94C9    E5 43      SBC $43        nein, maximale Nummer abziehen
94CB[1] 85 4D      STA $4D        und neue Sektornummer merken
94CD    A2 05      LDX #$05       Puffer 5 als
94CF    86 3F      STX $3F        aktuellen Puffer setzen
94D1    A2 FF      LDX #$FF       Wert für Pufferzeiger
94D3[1] 20 D1 93   JSR $93D1      Pufferzeiger setzen und Jobcode holen
94D6    10 43      BPL $951B      ist ein Jobcode vorhanden ?
94D8    29 01      AND #$01       ja, angesprochenes Laufwerk feststellen
94DA    C5 3E      CMP $3E        und mit aktuellem Laufwerk vergleichen
94DC    D0 3D      BNE $951B      identisch ?
94DE    A0 00      LDY #$00       ja, Zeiger auf Parameter von Puffer 0
94E0    B1 32      LDA ($32),Y    Spur des Jobs für Puffer 0 holen
94E2    C5 40      CMP $40        mit letzter Spur vergleichen
94E4    D0 35      BNE $951B      identisch ?
94E6    A5 45      LDA $45        ja, Befehlsbits des Jobcodes holen und
94E8    C9 60      CMP #$60       auf Kode für 'Programm ausführen' testen
94EA    F0 0C      BEQ $94F8      identisch ?
94EC    A0 01      LDY #$01       nein, Zeiger auf Parameter von Puffer 0
```

```
94EE   38        SEC           Sektornummer des Jobs
94EF   B1 32     LDA ($32),Y   für Puffer 1 holen
94F1   E5 4D     SBC $4D       auf errechnetem optimalem Sektor prüfen
94F3   10 03     BPL $94F8     ist neue Sektornummer kleiner ?
94F5   18        CLC           nein, Zahl der Sektoren bis zu diesem
94F6   65 43     ADC $43       Sektor berechnen
94F8[2] C5 4C    CMP $4C       und mit letzter Differenz vergleichen
94FA   B0 1F     BCS $951B     ist neuer Wert kleiner als letzter ?
94FC   48        PHA           ja, Sektordifferenz merken
94FD   A5 45     LDA $45       Befehlsbits des Jobcodes prüfen
94FF   F0 15     BEQ $9516     soll Sektor gelesen werden ?
9501   68        PLA           nein, Differenz wieder holen
9502   C9 09     CMP #$09      und mit 9 vergleichen
9504   90 15     BCC $951B     ist Wert kleiner ?
9506   C9 0C     CMP #$0C      nein, mit 13 vergleichen
9508   B0 11     BCS $951B     ist Differenz kleiner 13 ?
950A[1] 85 4C    STA $4C       ja, neue Sektordifferenz merken
950C   A5 3F     LDA $3F       Nummer des aktuellen Puffers
950E   AA        TAX           holen
950F   18        CLC           und daraus dann
9510   69 03     ADC #$03      die zugehörige physikalische
9512   85 31     STA $31       Speicheradresse berechnen
9514   D0 05     BNE $951B     immer Sprung nach $951B
9516[1] 68       PLA           Sektordifferenz wieder holen
9517   C9 06     CMP #$06      und mit 6 vergleichen
9519   90 EF     BCC $950A     ist Differenz kleiner ?
951B[7] C6 3F    DEC $3F       nein, Zeiger auf nächsten Puffer richten
951D   10 B4     BPL $94D3     schon alle Puffer geprüft ?
951F   8A        TXA           ja, Puffernummer des nächsten Jobs
9520   10 03     BPL $9525     optimaler Job gefunden ?
9522   4C CA 99  JMP $99CA     Stepperkommandos ausführen
9525[1] 86 3F    STX $3F       Nummer als aktuellen Puffer merken
9527   20 D1 93  JSR $93D1     Pufferzeiger setzen und Jobcode holen
952A   A5 45     LDA $45       Befehlbits des Jobcodes feststellen
952C   4C 06 96  JMP $9606     Lese- und Schreibjobs ausführen
-----------------------------------------------------------------
[9476] vgl. F497
Sektorheader von GCR nach Binär umrechnen
952F   A5 30     LDA $30       Low-Byte des aktuellen Pufferzeigers
9531   48        PHA           retten
9532   A5 31     LDA $31       High-Byte des aktuellen Pufferszeigers
```

```
9534  48        PHA         retten
9535  A9 24     LDA #$24    Pufferzeiger $30/$31
9537  85 30     STA $30     auf Beginn des Puffers
9539  A9 00     LDA #$00    für den aktuellen
953B  85 31     STA $31     Sektorheader setzen
953D  A9 00     LDA #$00    Pufferzeiger
953F  85 34     STA $34     zurücksetzen
9541  20 D9 98  JSR $98D9   5 GCR-Bytes in 4 Binärbytes umwandeln
9544  A5 55     LDA $55     erstes umgewandeltes Byte holen und
9546  85 18     STA $18     als Spurnummer des Headers merken
9548  A5 54     LDA $54     zweites umgewandeltes Byte holen
954A  85 19     STA $19     und als Sektornummer merken
954C  A5 53     LDA $53     drittes umgewandeltes Byte holen
954E  85 1A     STA $1A     und als Prüfsumme des Headers setzen
9550  20 D9 98  JSR $98D9   5 GCR-Bytes in 4 Binärbytes umwandeln
9553  A5 52     LDA $52     erstes umgewandeltes Byte
9555  85 17     STA $17     als zweites Zeichen der ID sezten
9557  A5 53     LDA $53     zweites umgewandeltes Binärbyte
9559  85 16     STA $16     als erstes Zeichen der ID setzen
955B  68        PLA         ursprünglichen Wert
955C  85 31     STA $31     des Pufferzeigers $30/$31
955E  68        PLA         wieder holen und
955F  85 30     STA $30     setzen
9561  60        RTS         zurück zur aufrufenden Routine
--------------------------------------------------------------------
9562  FF ...                unbenutzter
95FF  ... FF                ROM-Bereich
--------------------------------------------------------------------
[960D/98A6]  vgl. F50A
Datensektor suchen und auf Kopf auf Datenanfang setzen
9600  20 0F 97  JSR $970F   Sektorheader suchen
9603  4C 54 97  JMP $9754   nächste Sync-Markierung abwarten
--------------------------------------------------------------------
[952C]  vgl. F4CA
Comodore Sektor lesen, wenn Jobcode $80 (Befehlbits $00)
9606  C9 00     CMP #$00    Jobcode auf 'Sektor lesen' prüfen
9608  F0 03     BEQ $960D   identisch ?
960A  4C 6E 97  JMP $976E   nein, Jobcode weiter prüfen
--------------------------------------------------------------------
```

```
[92D9/9608]
Sektor lesen
960D  20 00 96   JSR $9600     Datenblock suchen
9610  2C 0F 18   BIT $180F     auf 'Byte Ready'-Signal
9613  30 FB      BMI $9610     warten
9615  AD 01 1C   LDA $1C01     Byte von Diskette lesen
9618  AA         TAX           und merken
9619  BD 0D A0   LDA $A00D,X   Binäraquivalent holen
961C  85 52      STA $52       und merken
961E  8A         TXA           ursprüngliches Byte wieder holen
961F  29 07      AND #$07      und ersten GCR-Teil ausblenden
9621  85 53      STA $53       Byte merken
9623  2C 0F 18   BIT $180F     auf 'Byte Ready'-Signal
9626  30 FB      BMI $9623     warten
9628  AD 01 1C   LDA $1C01     Byte von Diskette lesen
962B  85 54      STA $54       und merken
962D  29 C0      AND #$C0      letzte 2 Bits des 2. GCR-Bytes holen
962F  05 53      ORA $53       und erste 3 Bits einblenden
9631  AA         TAX           (1. Teil: Bit 0-2; 2. Teil: Bit 6-7)
9632  BD 0D 9F   LDA $9F0D,X   Binäräquivalent holen und mit vor-
9635  05 52      ORA $52       herigem Halbbyte erstes Binärbyte bilden
9637  48         PHA           Byte als Datenblockkennzeichen merken
9638  4C 67 96   JMP $9667     Datenteil einlesen
-----------------------------------------------------------------------
[963E/96D4]
GCR-Bytes von Diskette lesen und als Binärbytes im Puffer ablegen
963B  2C 0F 18   BIT $180F     auf 'Byte Ready'-Signal
963E  30 FB      BMI $963B     warten
9640  AD 01 1C   LDA $1C01     Byte von Diskette lesen
9643  AA         TAX           und merken
9644  BD 0D A0   LDA $A00D,X   Binäräquivalent feststellen
9647  85 52      STA $52       und zwischenspeichern
9649  8A         TXA           ursrüngliches Datenbyte wieder holen
964A  29 07      AND #$07      und erstes GCR-Byte ausblenden
964C  85 53      STA $53       Teil des 2. GCR-Bytes merken
964E  2C 0F 18   BIT $180F     auf 'Byte Ready'-Signal
9651  30 FB      BMI $964E     warten
9653  AD 01 1C   LDA $1C01     Byte von Diskette lesen
9656  85 54      STA $54       und merken
9658  29 C0      AND #$C0      letzten Teil des 2. GCR-Bytes holen
965A  05 53      ORA $53       und ersten Teil einblenden
```

```
965C  AA        TAX          (1. Teil: Bit 0-2; 2. Teil: Bit 6-7)
965D  BD 0D 9F  LDA $9F0D,X  entsprechendes Binärhalbbyte holen
9660  05 52     ORA $52      und vorheriges Halbbyte einblenden
9662  91 30     STA ($30),Y  Binärbyte in Puffer schreiben
9664  C8        INY          Pufferzeiger auf nächstes Byte richten
9665  F0 70     BEQ $96D7    Ende des Puffers erreicht ?
9667  A5 54     LDA $54      nein, nächstes GCR-Byte holen
9669  AA        TAX          und oberes Halbbyte des
966A  BD 0D A1  LDA $A10D,X  äquivalenten Binärbytes feststellen
966D  85 52     STA $52      und merken
966F  8A        TXA          ursprüngliches GCR-Byte wieder holen
9670  29 01     AND #$01     und erster Teil des nächsten GCR-Bytes
9672  85 54     STA $54      bilden und merken
9674  2C 0F 18  BIT $180F    auf 'Byte Ready'-Signal
9677  30 FB     BMI $9674    warten
9679  AD 01 1C  LDA $1C01    Byte von Diskette lesen
967C  85 55     STA $55      und merken
967E  29 F0     AND #$F0     zweiten Teil des GCR-Bytes herstellen
9680  05 54     ORA $54      und mit ersten Teil verknüpfen
9682  AA        TAX          (1. Teil: Bit0; 2. Teil: Bit 4-7)
9683  BD 0F 9F  LDA $9F0F,X  entsprechendes Binärhalbbyte holen
9686  05 52     ORA $52      und nächstes Binärbyte bilden
9688  91 30     STA ($30),Y  Byte in Puffer schreiben
968A  C8        INY          Pufferzeiger auf nächstes Byte richten
968B  A5 55     LDA $55      ersten Teil des nächsten
968D  29 0F     AND #$0F     GCR-Bytes herstellen
968F  85 55     STA $55      und merken
9691  2C 0F 18  BIT $180F    auf 'Byte Ready'-Signal
9694  30 FB     BMI $9691    warten
9696  AD 01 1C  LDA $1C01    Byte von Diskette lesen
9699  85 3A     STA $3A      und merken
969B  29 80     AND #$80     zweiten Teil des GCR-Bytes erstellen
969D  05 55     ORA $55      und mit erstem Teil verknüpfen
969F  AA        TAX          (1. Teil: Bit 0-3; 2. Teil: Bit7)
96A0  BD 1D 9F  LDA $9F1D,X  erstes Halbbyte des nächsten Binärwerts
96A3  85 52     STA $52      feststellen und zwischenspeichern
96A5  A5 3A     LDA $3A      ursprünglichen GCR-Wert wieder holen
96A7  AA        TAX          und zweites Halbbyte
96A8  BD 0D A2  LDA $A20D,X  des äuivalenten Binärbytes holen
96AB  05 52     ORA $52      ersten Teil einblenden und
96AD  91 30     STA ($30),Y  Byte in Puffer schreiben
```

```
96AF  C8        INY            Pufferzeiger auf nächstes Byte richten
96B0  8A        TXA            ersten Teil des
96B1  29 03     AND #$03       nächsten GCR-Bytes herstellen
96B3  85 3A     STA $3A        und merken
96B5[1] 2C 0F 18  BIT $180F    auf 'Byte Ready'-Signal
96B8  30 FB     BMI $96B5      warten
96BA  AD 01 1C  LDA $1C01      Byte von Diskette lesen
96BD  85 53     STA $53        und zwischenspeichern
96BF  29 E0     AND #$E0       zweiten Teil des GCR-Bytes isolieren
96C1  05 3A     ORA $3A        und mit erstem Teil verknüpfen
96C3  AA        TAX            (1. Teil: Bit 0-1; 2. Teil: Bit 5-7)
96C4  BD 2A 9F  LDA $9F2A,X    erstes Binärhalbbyte feststellen
96C7  85 52     STA $52        und merken
96C9  A5 53     LDA $53        ursprünglichen GCR-Wert wieder holen
96CB  AA        TAX            und zweiten Teil des
96CC  BD 0D A3  LDA $A30D,X    Binärbytes holen
96CF  05 52     ORA $52        erstes Halbbyte einblenden
96D1  91 30     STA ($30),Y    Byte in Puffer schreiben
96D3  C8        INY            Pufferzeiger auf nächste Position setzen
96D4  4C 3B 96  JMP $963B      nächste 5 GCR-Bytes in 4 Binärbytes
-----------------------------------------------------------------
[9665]  Ende des Puffers erreicht
96D7  A5 54     LDA $54        letztes GCR-Byte holen
96D9  AA        TAX            und erstes Halbbyte des
96DA  BD 0D A1  LDA $A10D,X    nächsten Binärbytes feststellen
96DD  85 52     STA $52        und merken
96DF  8A        TXA            ursprünglichen GCR-Wert wieder holen
96E0  29 01     AND #$01       und ersten Teil des nächsten GCR-Bytes
96E2  85 54     STA $54        isolieren
96E4[1] 2C 0F 18  BIT $180F    auf 'Byte Ready'-Signal
96E7  30 FB     BMI $96E4      warten
96E9  AD 01 1C  LDA $1C01      Byte von Diskette lesen
96EC  29 F0     AND #$F0       und zweiten Teil des GCR-Bytes holen
96EE  05 54     ORA $54        mit erstem Teil verknüpfen
96F0  AA        TAX            (1. Teil: Bit0; 2. Teil: Bit 4-7)
96F1  BD 0F 9F  LDA $9F0F,X    zweiten Teil des Binärbytes feststellen
96F4  05 52     ORA $52        und endgültiges Binärbyte bilden
96F6  85 53     STA $53        Wert als Prüfsumme merken
96F8  68        PLA            Datenblockkennzeichen wieder holen
96F9  C5 47     CMP $47        und prüfen
96FB  D0 0A     BNE $9707      ist Kennzeichen richtig ?
```

```
96FD  20 E9 F5   JSR $F5E9      ja, Prüfsumme des Puffers berechnen
9700  C5 53      CMP $53        auf angegebener Prüfsumme prüfen
9702  F0 06      BEQ $970A      ist ein Lesefehler aufgetreten ?
9704  A9 05      LDA #$05       ja, Fehlernummer für 'Read Error (23)'
9706  2C         .byte $2C      nächste 2 Bytes überspringen (Bitbefehl)
9707¹ A9 04      LDA #$04       Fehlernummer für 'Read Error (22)'
9709  2C         .byte $2C      nächste 2 Bytes überspringen (Bitbefehl)
970A¹ A9 01      LDA #$01       Wert für 'Ok' Meldung
970C  4C B5 99   JMP $99B5      Rückmeldung übergeben
---------------------------------------------------------------------
[9600/9789/98CE]  vgl. F510
Sektorheader suchen
970F  A5 12      LDA $12        aktuelle ID (erstes Zeichen)
9711  85 16      STA $16        in Puffer für Sektorheader schreiben
9713  A5 13      LDA $13        aktuelle ID (zweites Zeichen)
9715  85 17      STA $17        in Puffer für Sektorheader schreiben
9717  A0 00      LDY #$00       Pufferzeiger zurücksetzen
9719  B1 32      LDA ($32),Y    Spur des aktuellen Jobs holen und
971B  85 18      STA $18        in Headerpuffer übernehmen
971D  C8         INY            Pufferzeiger auf nächstes Byte
971E  B1 32      LDA ($32),Y    Nummer des aktuellen Sektors
9720  85 19      STA $19        in Header übernehmen
9722  A9 00      LDA #$00       Prüfsumme errechnen :
9724  45 16      EOR $16        ID 1
9726  45 17      EOR $17        ID 2
9728  45 18      EOR $18        Spurnummer
972A  45 19      EOR $19        Sektornummer
972C  85 1A      STA $1A        Prüfsumme in Headerpuffer
972E  20 34 F9   JSR $F934      Header in GCR-Werte umrechnen
9731  A9 5A      LDA #$5A       Zahl der Leseversuche (90)
9733  85 4B      STA $4B        setzen
9735¹ 20 54 97   JSR $9754      auf nächste Sync-Markierung warten
9738¹ B9 24 00   LDA $0024,Y    Byte aus Headerpuffer holen
973B¹ 2C 0F 18   BIT $180F      auf 'Byte Ready'-Signal
973E  30 FB      BMI $973B      warten
9740  CD 01 1C   CMP $1C01      mit Byte auf Diskette vergleichen
9743  D0 06      BNE $974B      identisch ?
9745  C8         INY            ja, nächstes Byte vergleichen
9746  C0 08      CPY #$08       Zahl der Bytes des Headers
9748  D0 EE      BNE $9738      gesamter Header schon verglichen ?
974A  60         RTS            ja, zurück zur aufrufenden Routine
```

```
974B[1] C6 4B      DEC $4B        noch ein Versuch
974D    D0 E6      BNE $9735      Zahl der Leseversuche zu Ende ?
974F    A9 02      LDA #$02       ja, Fehlernummer für 'Read Error (21)'
9751    4C B5 99   JMP $99B5      Rückmeldung übergeben
---------------------------------------------------------------------
[9453/9603/9735/9CDD/9D08/BF21]  vgl. F556
auf nächste Sync-Markierung warten
9754    A2 0F      LDX #$0F       Zähler für Versuche setzen
9756    A0 00      LDY #$00       (ca. 47 / 23 ms suchen)
9758    2C 00 1C   BIT $1C00      'Sync'-Signal prüfen
975B    10 0B      BPL $9768      ist Sync gesetzt ?
975D    88         DEY            nein, Zähler erniedrigen
975E    D0 F8      BNE $9758      Zähler schon abgelaufen
9760    CA         DEX            Zähler erniedrigen
9761    D0 F5      BNE $9758      schon 256 Testzyklen abgelaufen ?
9763    A9 03      LDA #$03       Fehlernummer für 'sync not found'
9765    4C B5 99   JMP $99B5      Rückmeldung übergeben
9768    AD 01 1C   LDA $1C01      'Byte Ready' (CA1) initialisieren
976B    A0 00      LDY #$00       Register löschen
976D    60         RTS            zurück zur aufrufenden Routine
---------------------------------------------------------------------
[960A]  vgl. F56E
Sektor schreiben, wenn Jobcode $90 (Befehlbits $10)
976E    C9 10      CMP #$10       auf Jobcode für 'Schreiben' prüfen
9770    F0 03      BEQ $9775      soll Sektor geschrieben werden ?
9772    4C 98 98   JMP $9898      nein, Jobcode weiter prüfen
---------------------------------------------------------------------
Sektor schreiben
9775[1] 20 E9 F5   JSR $F5E9      Prüfsumme des Puffers errechnen
9778    85 3A      STA $3A        und merken
977A    AD 00 1C   LDA $1C00      Laufwerkssteuerregister holen
977D    29 10      AND #$10       und Bit für 'Write Protect' prüfen
977F    D0 05      BNE $9786      ist Schreibschutz gesetzt ?
9781    A9 08      LDA #$08       ja, Fehlernummer für 'write protect on'
9783    4C B5 99   JMP $99B5      Rückmeldung setzen
9786[1] 20 8F F7   JSR $F78F      Pufferinhalt in GCR-Werte umrechnen
9789    20 0F 97   JSR $970F      Blockheader suchen
978C    A0 09      LDY #$09       Zahl der Bytes für Lücke bis Datenblock
978E[2] 2C 0F 18   BIT $180F      auf 'Byte Ready'-Signal
9791    30 FB      BMI $978E      warten
9793    2C 00 1C   BIT $1C00      Kopf wieder bereit machen
```

```
9796      88          DEY            noch ein Byte
9797      D0 F5       BNE $978E      Lücke schon übersprungen ?
9799      A9 FF       LDA #$FF       ja, Kopfregister auf Ausgang
979B      8D 03 1C    STA $1C03      umstellen
979E      AD 0C 1C    LDA $1C0C      Steuerregister holen
97A1      29 1F       AND #$1F       und Kopfelektronik
97A3      09 C0       ORA #$C0       auf Schreiben stellen
97A5      8D 0C 1C    STA $1C0C      (CB2 auf Low)
97A8      A9 FF       LDA #$FF       Wert für Sync-Markierung
97AA      A0 05       LDY #$05       Zahl der Sync-Bytes
97AC      8D 01 1C    STA $1C01      Byte auf Diskette schreiben
97AF[2]   2C 0F 18    BIT $180F      auf 'Byte Ready'-Signal
97B2      30 FB       BMI $97AF      warten
97B4      2C 00 1C    BIT $1C00      'Byte Ready' zurücksetzen
97B7      88          DEY            nächstes Sync-Byte
97B8      D0 F5       BNE $97AF      gesamte Markierung schon geschrieben ?
97BA      A0 BB       LDY #$BB       ja, Pufferzeiger auf Zusatzpuffer
97BC[1]   B9 00 01    LDA $0100,Y    Byte aus Puffer holen
97BF[1]   2C 0F 18    BIT $180F      auf 'Byte Ready'-Signal
97C2      30 FB       BMI $97BF      warten
97C4      8D 01 1C    STA $1C01      GCR-Byte auf Diskette schreiben
97C7      C8          INY            Zeiger auf nächstes Byte richten
97C8      D0 F2       BNE $97BC      ganzer Zusatzpuffer auf Diskette ?
97CA[1]   B1 30       LDA ($30),Y    ja, Byte aus aktuellem Puffer holen
97CC[1]   2C 0F 18    BIT $180F      auf 'Byte Ready'-Signal
97CF      30 FB       BMI $97CC      warten
97D1      8D 01 1C    STA $1C01      GCR-Byte auf Diskette schreiben
97D4      C8          INY            Pufferzeiger auf nächstes Byte setzen
97D5      D0 F3       BNE $97CA      ganzer Puffer geschrieben ?
97D7[1]   2C 0F 18    BIT $180F      auf 'Byte Ready'-Signal
97DA      30 FB       BMI $97D7      warten, bis Byte geschrieben
97DC      AD 0C 1C    LDA $1C0C      Kopfelektronik
97DF      09 E0       ORA #$E0       auf Lesen umschalten
97E1      8D 0C 1C    STA $1C0C      (CB2 auf High)
97E4      A9 00       LDA #$00       Kopfregister auf
97E6      8D 03 1C    STA $1C03      Eingang stellen
97E9      20 F9 97    JSR $97F9      Puffer von GCR nach Binär zurückwandeln
97EC      A4 3F       LDY $3F        Nummer des aktuellen Puffers
97EE      B9 00 00    LDA $0000,Y    Jobcode holen
97F1      49 30       EOR #$30       und in 'Sektor überprüfen' umwandeln
97F3      99 00 00    STA $0000,Y    neuen Jobcode setzen
```

```
97F6  4C 4F 94   JMP $944F         Header des Sektors suchen
-----------------------------------------------------------------
[97E9/99BE]  vgl. F5F2
aktuellen Puffer und Zusatzpuffer ($01BB-$1FF) von GCR nach Binär wandeln
97F9  A9 00      LDA #$00          Low-Byte des Zeigers für den
97FB  85 2E      STA $2E           aktuellen Datenpuffer und den
97FD  85 30      STA $30           Zusatzpuffer initialisieren
97FF  85 4F      STA $4F           momentanen Wert des
9801  A5 31      LDA $31           Zeigers auf den aktuellen Datenpuffer
9803  85 4E      STA $4E           in $4E/$4F retten
9805  A9 01      LDA #$01          Pufferzeiger auf
9807  85 31      STA $31           Zusatzpuffer richten
9809  85 2F      STA $2F           (High-Byte)
980B  A9 BB      LDA #$BB          Pufferzeiger für Umwandlungsroutine auf
980D  85 34      STA $34           Beginn des Zusatzpuffers richten ($1BB)
980F  85 36      STA $36           Zeiger auf akt. Binärbyteposition setzen
9811  20 D9 98   JSR $98D9         5 GCR-Bytes in 4 Binärbytes umwandeln
9814  A5 52      LDA $52           Kennzeichen des Datenblocks holen
9816  85 38      STA $38           und merken
9818  A4 36      LDY $36           Pufferzeiger auf nächstes Binärbyte
holen
981A  A5 53      LDA $53           Datenbyte holen
981C  91 2E      STA ($2E),Y       und in Puffer schreiben
981E  C8         INY               Pufferzeiger auf nächstes Byte setzen
981F  A5 54      LDA $54           Datenbyte holen
9821  91 2E      STA ($2E),Y       und in Puffer schreiben
9823  C8         INY               Pufferzeiger auf nächstes Byte setzen
9824  A5 55      LDA $55           Datenbyte holen
9826  91 2E      STA ($2E),Y       und in Puffer schreiben
9828  C8         INY               Pufferzeiger auf nächstes Byte setzen
9829  84 36      STY $36           Pufferzeiger merken
982B  20 D9 98   JSR $98D9         5 GCR-Bytes in 4 Binärbytes umrechnen
982E  A4 36      LDY $36           Pufferzeiger wieder holen
9830  A5 52      LDA $52           Datenbyte holen
9832  91 2E      STA ($2E),Y       und in Puffer schreiben
9834  C8         INY               Pufferzeiger auf nächstes Byte setzen
9835  A5 53      LDA $53           Datenbyte holen
9837  91 2E      STA ($2E),Y       und in Puffer schreiben
9839  C8         INY               Pufferzeiger auf nächstes Byte setzen
983A  F0 0E      BEQ $984A         Ende des Zusatzpuffers erreicht ?
983C  A5 54      LDA $54           nein, Datenbyte holen
```

```
983E    91 2E      STA ($2E),Y    und in Puffer schreiben
9840    C8         INY            Pufferzeiger auf nächstes Byte setzen
9841    A5 55      LDA $55        Datenbyte holen
9843    91 2E      STA ($2E),Y    und in Puffer schreiben
9845    C8         INY            Pufferzeiger auf nächstes Byte setzen
9846    84 36      STY $36        Pufferzeiger merken
9848    D0 E1      BNE $982B      Ende des Zusatzpuffers erreicht ?
984A[1] A5 54      LDA $54        ja, umgewandeltes Binärbyte holen und
984C    91 30      STA ($30),Y    in Datenpuffer schreiben
984E    C8         INY            Pufferzeiger auf nächstes Byte setzen
984F    A5 55      LDA $55        umgewandeltes Binärbyte holen und
9851    91 30      STA ($30),Y    in Datenpuffer schreiben
9853    C8         INY            Pufferzeiger auf nächstes Byte setzen
9854    84 36      STY $36        Pufferzeiger merken
9856[1] 20 D9 98   JSR $98D9      5 GCR-Bytes in 4 Binärbytes umwandeln
9859    A4 36      LDY $36        Pufferzeiger wieder holen
985B    A5 52      LDA $52        umgewandeltes Binärbyte holen und
985D    91 30      STA ($30),Y    in Datenpuffer schreiben
985F    C8         INY            Pufferzeiger auf nächstes Byte setzen
9860    A5 53      LDA $53        umgewandeltes Binärbyte holen und
9862    91 30      STA ($30),Y    in Datenpuffer schreiben
9864    C8         INY            Pufferzeiger auf nächstes Byte setzen
9865    A5 54      LDA $54        umgewandeltes Binärbyte holen und
9867    91 30      STA ($30),Y    in Datenpuffer schreiben
9869    C8         INY            Pufferzeiger auf nächstes Byte setzen
986A    A5 55      LDA $55        umgewandeltes Binärbyte holen und
986C    91 30      STA ($30),Y    in Datenpuffer schreiben
986E    C8         INY            Pufferzeiger auf nächstes Byte setzen
986F    84 36      STY $36        Pufferzeiger merken und
9871    C0 BB      CPY #$BB       mit Endwert vergleichen
9873    90 E1      BCC $9856      Ende des Datenpuffers erreicht ?
9875    A9 45      LDA #$45       ja, Zeiger auf
9877    85 2E      STA $2E        Zieladresse der
9879    A5 31      LDA $31        nachfolgenden Verschiebeaktion
987B    85 2F      STA $2F        setzen
987D    A0 BA      LDY #$BA       Bytes in Datenpuffer von Position
987F[1] B1 30      LDA ($30),Y    $01-$BB nach Position $46-$FF im Puffer
9881    91 2E      STA ($2E),Y    nach oben verschieben
9883    88         DEY            Pufferzeiger auf nächstes Byte
9884    D0 F9      BNE $987F      schon alle Zeichen verschoben ?
9886    B1 30      LDA ($30),Y    ja, Byte $00 nach
```

```
9888  91 2E     STA ($2E),Y    Position $45 kopieren
988A  A2 BB     LDX #$BB       Zeiger auf Beginn des Zusatzpuffers
988C  BD 00 01  LDA $0100,X    Byte aus Zusatzpuffer holen
988F  91 30     STA ($30),Y    und in Datenpuffer kopieren
9891  C8        INY            Pufferzeiger für Daten- und
9892  E8        INX            Zusatzpuffer auf nächstes Byte richten
9893  D0 F7     BNE $988C      schon ganzer Puffer kopiert ?
9895  86 50     STX $50        ja, Flag für 'Puffer in GCR' löschen (0)
9897  60        RTS            zurück zur aufrufenden Routine
-----------------------------------------------------------------------
[9772]  vgl. F691
Sektor überprüfen (Verify), wenn Jobcode $A0 ist (Befehlbits $20)
9898  C9 20     CMP #$20       auf Jobcode für 'Sektor prüfen' testen
989A  F0 02     BEQ $989E      soll Sektor verifiziert werden ?
989C  D0 30     BNE $98CE      nein, immer Sprung nach $98CE
-----------------------------------------------------------------------
Sektor verify
989E  20 E9 F5  JSR $F5E9      Prüfsumme des Puffers errechnen
98A1  85 3A     STA $3A        und merken
98A3  20 8F F7  JSR $F78F      Puffer von Binär nach GCR umwandeln
98A6  20 00 96  JSR $9600      Sektorheader suchen
98A9  A0 BB     LDY #$BB       Zeiger auf Zusatzpuffer richten
98AB  B9 00 01  LDA $0100,Y    Byte aus Puffer holen
98AE  2C 0F 18  BIT $180F      auf 'Byte Ready'-Signal
98B1  30 FB     BMI $98AE      warten
98B3  4D 01 1C  EOR $1C01      und mit Byte von Diskette vergleichen
98B6  D0 1C     BNE $98D4      identisch ?
98B8  C8        INY            ja, Pufferzeiger auf nächstes Byte
98B9  D0 F0     BNE $98AB      ganzer Zusatzpuffer verglichen ?
98BB  B1 30     LDA ($30),Y    Byte aus Datenpuffer
98BD  2C 0F 18  BIT $180F      auf 'Byte Ready'-Signal
98C0  30 FB     BMI $98BD      warten
98C2  4D 01 1C  EOR $1C01      mit Byte von Diskette vergleichen
98C5  D0 0D     BNE $98D4      identisch ?
98C7  C8        INY            ja, Pufferzeiger auf nächstes Byte
98C8  C0 FD     CPY #$FD       Zeiger mit Endwert vergleichen
98CA  D0 EF     BNE $98BB      Ende des Datenpuffers erreicht ?
98CC  F0 03     BEQ $98D1      ja, immer Sprung nach $98D1
98CE  20 0F 97  JSR $970F      nächsten Sektorheader suchen
98D1  A9 01     LDA #$01       Nummer der 'Ok' Meldung
98D3  2C        .byte $2C      nächste 2 Bytes überspringen
```

```
98D4[2] A9 07     LDA #$07      Fehlernummer für 'Verify Error' setzen
98D6 4C B5 99     JMP $99B5     Rückmeldung übergeben
---------------------------------------------------------------------

[9541/9550/9811/982B/9856/9979/9993/BF30]  vgl. F7E6
5 GCR-Bytes in 4 Binärbytes umrechnen
98D9  A4 34       LDY $34       Zeiger auf nächstes GCR-Byte holen
98DB  B1 30       LDA ($30),Y   GCR-Byte aus Puffer holen
98DD  85 56       STA $56       und ersten GCR-Wert merken
98DF  29 07       AND #$07      1. Teil des 2. GCR-Werts
98E1  85 57       STA $57       merken
98E3  C8          INY           Zeiger auf nächstes GCR-Byte
98E4  D0 06       BNE $98EC     Ende des Zusatzpuffers erreicht ?
98E6  A5 4E       LDA $4E       ja, Zeiger auf Beginn des
98E8  85 31       STA $31       aktuellen Datenpuffers setzen
98EA  A4 4F       LDY $4F       Zeiger auf Position in Puffer setzen
98EC[1] B1 30     LDA ($30),Y   GCR-Byte aus Puffer holen
98EE  85 58       STA $58       und merken
98F0  29 C0       AND #$C0      2. Teil des 2. GCR-Werts holen
98F2  05 57       ORA $57       und ersten Teil einblenden
98F4  85 57       STA $57       zweiten GCR-Wert merken
98F6  A5 58       LDA $58       orginal GCR-Byte nochmal holen
98F8  29 01       AND #$01      und 1. Teil des 3. GCR-Werts holen
98FA  85 59       STA $59       Wert merken
98FC  C8          INY           Pufferzeiger auf nächstes Byte setzen
98FD  B1 30       LDA ($30),Y   Byte aus Puffer holen
98FF  AA          TAX           und merken
9900  29 F0       AND #$F0      2. Teil des 3. GCR-Werts herstellen
9902  05 59       ORA $59       und mit erstem Teil verknüpfen
9904  85 59       STA $59       gesamtes Byte merken
9906  8A          TXA           orginal GCR-Byte nochmal holen
9907  29 0F       AND #$0F      und 1. Teil des 4. GCR-Werts
9909  85 5A       STA $5A       merken
990B  C8          INY           Pufferzeiger auf nächstes Byte richten
990C  B1 30       LDA ($30),Y   Byte aus Puffer holen
990E  85 5B       STA $5B       und merken
9910  29 80       AND #$80      2. Teil des 4. GCR-Werts holen
9912  05 5A       ORA $5A       und vorherigen ersten Teil einblenden
9914  85 5A       STA $5A       gesamten Wert merken
9916  A5 5B       LDA $5B       orginal GCR-Byte nochmal holen
9918  29 03       AND #$03      und 1. Teil des 5.GCR-Werts
991A  85 5C       STA $5C       isolieren
```

```
991C  C8        INY            Zeiger auf nächstes Byte setzen
991D  D0 08     BNE $9927      Ende des Zusatzpuffers erreicht ?
991F  A5 4E     LDA $4E        ja, Zeiger auf aktuellen
9921  85 31     STA $31        Datenpuffer richten
9923  A4 4F     LDY $4F        Zeiger auf Position in Puffer
9925  84 30     STY $30        setzen
9927[1] B1 30   LDA ($30),Y    Byte aus Puffer holen
9929  85 5D     STA $5D        und merken
992B  29 E0     AND #$E0       2. Teil des 5. GCR-Werts holen
992D  05 5C     ORA $5C        und ersten Teil einblenden
992F  85 5C     STA $5C        gesamten GCR-Wert merken
9931  C8        INY            Pufferzeiger auf nächstes Zeichen
9932  84 34     STY $34        und merken
9934  A6 56     LDX $56        ersten GCR-Wert holen und äquivalentes
9936  BD 0D A0  LDA $A00D,X    höherwertiges Binärhalbbyte feststellen
9939  A6 57     LDX $57        zweiten GCR-Wert holen und
993B  1D 0D 9F  ORA $9F0D,X    niederwertiges Binärhalbbyte bilden
993E  85 52     STA $52        erstes umgewandeltes Binärbyte merken
9940  A6 58     LDX $58        dritten GCR-Wert holen und äquivalentes
9942  BD 0D A1  LDA $A10D,X    höherwertiges Binärhalbbyte feststellen
9945  A6 59     LDX $59        vierten GCR-Wert holen und
9947  1D 0F 9F  ORA $9F0F,X    niederwertiges Binärhalbbyte bilden
994A  85 53     STA $53        zweites umgewandeltes Binärbyte merken
994C  A6 5A     LDX $5A        fünften GCR-Wert holen und äquivalentes
994E  BD 1D 9F  LDA $9F1D,X    höherwertiges Binärhalbbyte feststellen
9951  A6 5B     LDX $5B        sechsten GCR-Wert holen und
9953  1D 0D A2  ORA $A20D,X    niederwertiges Binärhalbbyte bilden
9956  85 54     STA $54        drittes umgewandeltes Binärbyte merken
9958  A6 5C     LDX $5C        siebten GCR-Wert holen und äquivalentes
995A  BD 2A 9F  LDA $9F2A,X    höherwertiges Binärhalbbyte feststellen
995D  A6 5D     LDX $5D        achten  GCR-Wert holen und
995F  1D 0D A3  ORA $A30D,X    niederwertiges Binärhalbbyte bilden
9962  85 55     STA $55        letztes umgewandeltes Binärbyte merken
9964  60        RTS            zurück zur aufrufenden Routine
-----------------------------------------------------------------------

[BF27/Routine wird im DOS nicht benutzt]  vgl. F8E0
Zusatztpuffer von GCR nach Binär umwandeln
9965  A9 00     LDA #$00       Zeiger auf akt. GCR-Byte
9967  85 34     STA $34        zurücksetzen
9969  85 2E     STA $2E        Zeiger auf Zielpuffer löschen
996B  85 36     STA $36        Zeiger auf akt. Datenposition
```

```
996D  A9 01     LDA #$01        Zwischenspeicher für
996F  85 4E     STA $4E         Adresse des aktuellen Datenpuffers
9971  A9 BA     LDA #$BA        auf Beginn des Zusatzpuffers
9973  85 4F     STA $4F         richten
9975  A5 31     LDA $31         Pufferzeiger auf Wert des aktuellen
9977  85 2F     STA $2F         Datenpuffers setzen
9979  20 D9 98  JSR $98D9       5 GCR-Bytes in 4 Binärbytes wandeln
997C  A5 52     LDA $52         erstes Binärbyte holen und als
997E  85 38     STA $38         Kennzeichen des Datenblocks übernehmen
9980  A4 36     LDY $36         Zeiger auf akt. Byte
9982  A5 53     LDA $53         zweites umgewandeltes Byte holen
9984  91 2E     STA ($2E),Y     und in Puffer schreiben
9986  C8        INY             Pufferzeiger auf nächstes Byte
9987  A5 54     LDA $54         drittes umgewandeltes Byte holen
9989  91 2E     STA ($2E),Y     und in Puffer schreiben
998B  C8        INY             Pufferzeiger auf nächstes Byte
998C  A5 55     LDA $55         letztes umgewandeltes Byte holen
998E  91 2E     STA ($2E),Y     und in Puffer schreiben
9990  C8        INY             Pufferzeiger auf nächstes Byte
9991  84 36     STY $36         und merken
9993  20 D9 98  JSR $98D9       5 GCR-Bytes in 4 Binärbytes wandeln
9996  A4 36     LDY $36         Pufferzeiger wieder holen
9998  A5 52     LDA $52         erstes umgewandeltes Byte holen
999A  91 2E     STA ($2E),Y     und in Puffer schreiben
999C  C8        INY             Pufferzeiger auf nächstes Byte
999D  F0 11     BEQ $99B0       Ende des Puffers erreicht ?
999F  A5 53     LDA $53         nein, zweites umgewandeltes Binärbyte
99A1  91 2E     STA ($2E),Y     holen und in Puffer schreiben
99A3  C8        INY             Pufferzeiger auf nächstes Byte setzen
99A4  A5 54     LDA $54         drittes umgewandeltes Byte holen
99A6  91 2E     STA ($2E),Y     und in Puffer schreiben
99A8  C8        INY             Pufferzeiger auf nächstes Byte
99A9  A5 55     LDA $55         drittes umgewandeltes Byte holen
99AB  91 2E     STA ($2E),Y     und in Puffer schreiben
99AD  C8        INY             Pufferzeiger auf nächstes Byte
99AE  D0 E1     BNE $9991       Ende des Puffers erreicht ?
99B0  A5 2F     LDA $2F         ja, Zeiger auf aktuellen Datenpuffer
99B2  85 31     STA $31         wieder herstellen
99B4  60        RTS             zurück zur aufrufenden Routine
--------------------------------------------------------------------
```

```
[92EB/93CE/94A6/94B9/970C/9751/9765/9783/9806/904E/9D60/BF15]  vgl. F969
Rückmeldung der Jobschleife übergeben
99B5  A4 3F     LDY $3F        Nummer des aktuellen Puffers
99B7  99 00 00  STA $0000,Y    Rückmeldung in Jobregister schreiben
99BA  A5 50     LDA $50        Flag für 'Puffer in GCR-Code'
99BC  F0 03     BEQ $99C1      ist der Puffer noch in GCR ?
99BE  20 F9 97  JSR $97F9      ja, Puffer von GCR nach Binär wandeln
99C1[1]  20 8F F9  JSR $F98F   Laufwerksmotor ausschalten
99C4  A6 49     LDX $49        Stapelzeiger wieder
99C6  9A        TXS            einrichten
99C7  4C C8 92  JMP $92C8      1571 Jobschleife
-------------------------------------------------------------------------
[92D2/92F8/9302/9329/9522/9B55/9B64/9D41/9D56/BF72]
Teil der Jobschleife für Motor- und Steppersteuerung
99CA  AD 07 1C  LDA $1C07      Timer 1 (High-Byte)
99CD  8D 05 1C  STA $1C05      wieder setzen
99D0  AD 00 1C  LDA $1C00      Laufwerkssteuerregister holen
99D3  29 10     AND #$10       und 'Write Protect' prüfen
99D5  C5 1E     CMP $1E        mit letzter Prüfung vergleichen
99D7  85 1E     STA $1E        und aktuellen Zustand merken
99D9  D0 07     BNE $99E2      hat sich der 'Write Protect' geändert ?
99DB  AD AB 02  LDA $02AB      nein, Zähler für Motorlaufzeit
99DE  D0 10     BNE $99F0      ist Motor angeschaltet ?
99E0  F0 1C     BEQ $99FE      nein, immer Sprung nach $99FE
99E2[1]  A9 FF     LDA #$FF    Zähler für Motorlaufzeit bei Diskwechsel
99E4  8D AB 02  STA $02AB      setzen
99E7  20 64 87  JSR $8764      Motor ausschalten
99EA  A9 01     LDA #$01       Flag für 'Disketten neu initialisieren'
99EC  85 1C     STA $1C        setzen
99EE  D0 0E     BNE $99FE      immer Sprung nach $99FE
99F0[1]  CE AB 02  DEC $02AB   Zähler für Motorlaufzeit erniedrigen
99F3  D0 09     BNE $99FE      Motor jetzt ausschalten ?
99F5  A5 20     LDA $20        ja, Laufwerksstatus holen
99F7  C9 00     CMP #$00       mit Wert für 'Motor aus' vergleichen
99F9  D0 03     BNE $99FE      identisch ?
99FB  20 70 87  JSR $8770      ja, Laufwerksmotor ausschalten
99FE[4]  AD FE 02  LDA $02FE   Steuerbyte für Kopf bei Lesefehlern
9A01  F0 15     BEQ $9A18      soll Kopf neben Spur gesetzt werden ?
9A03  C9 02     CMP #$02       nein, auf 'Steuerbyte angenommen' prüfen
9A05  D0 07     BNE $9A0E      ist Kopf gerade gesetzt worden ?
9A07  A9 00     LDA #$00       Steuerbyteregister
```

```
9A09     8D FE 02   STA $02FE     löschen
9A0C     F0 0A      BEQ $9A18     immer Sprung nach $9A18
9A0E[1]  85 4A      STA $4A       Zahl der auszuführenden Schritte setzen
9A10     A9 02      LDA #$02      Flag für 'Steuerbyte übernommen'
9A12     8D FE 02   STA $02FE     setzen
9A15     4C 56 9A   JMP $9A56     Kopf weiter positionieren
9A18[2]  A6 3E      LDX $3E       Flag für Laufwerk aktiv
9A1A     30 07      BMI $9A23     ist Flag gesetzt ?
9A1C     A5 20      LDA $20       nein, Laufwerksstatus holen
9A1E     A8         TAY           und merken
9A1F     C9 20      CMP #$20      mit Flag für 'Motor an' vergleichen
9A21     D0 03      BNE $9A26     ist Laufwerk bereit ?
9A23[2]  4C C9 9A   JMP $9AC9     ja, zurück zur aufrufenden Routine
9A26[1]  C6 48      DEC $48       Zähler für Anlaufverzögerung des Motors
9A28     D0 1C      BNE $9A46     ist Motor auf Drehzahl ?
9A2A     98         TYA           ja, Laufwerksstatus holen
9A2B     10 04      BPL $9A31     Flag für 'Motor nicht bereit' gesetzt ?
9A2D     29 7F      AND #$7F      ja, dann Flag
9A2F     85 20      STA $20       löschen
9A31[1]  29 10      AND #$10      Flag für 'Motor in Ausschaltphase'
9A33     F0 11      BEQ $9A46     soll Motor ausgeschalten werden ?
9A35     C6 35      DEC $35       noch auszuführende Jobschleifenaufrufe
9A37     D0 0D      BNE $9A46     Jobschleife nochmal aufrufen ?
9A39     20 70 87   JSR $8770     nein, Laufwerksmotor ausschalten
9A3C     A9 FF      LDA #$FF      Flag für 'Laufwerk aktiv'
9A3E     85 3E      STA $3E       löschen
9A40     A9 00      LDA #$00      Laufwerksstatus
9A42     85 20      STA $20       zurücksetzen
9A44     F0 DD      BEQ $9A23     immer Sprung nach $9A23
9A46[3]  98         TYA           Laufwerksstatus wieder holen
9A47     29 40      AND #$40      Flag für 'Stepper in Betrieb' testen
9A49     D0 03      BNE $9A4E     wird Kopf bewegt ?
9A4B     4C C9 9A   JMP $9AC9     nein, zurück zur aufrufenden Routine
9A4E[1]  A5 62      LDA $62       Flag für aktuelle Stepperphase
9A50     D0 50      BNE $9AA2     ist Kopf positioniert ?
9A52     A5 4A      LDA $4A       nein, Zahl der zu fahrenden Schritte
9A54     F0 43      BEQ $9A99     Zähler gesetzt ?
```

---3007-

```
[9A15]
Kopfsteuerroutine
9A56  A5 4A     LDA $4A     ja, Zahl der Halbschritte holen
9A58  10 59     BPL $9AB3   soll Kopf nach außen bewegt werden ?
9A5A  98        TYA         ja, Laufwerksstatus holen
9A5B  48        PHA         und retten
9A5C  A0 63     LDY #$63    Zahl der Abtastversuche (99)
9A5E  AD 0F 18  LDA $180F   Steuerregister A holen und Zustand der
9A61  6A        ROR A       Spur0-Lichtschranke ins Carry setzen
9A62  08        PHP         Carry merken
9A63  AD 0F 18  LDA $180F   Steuerregister nochmal lesen und
9A66  6A        ROR A       Lichtschranke erneut testen
9A67  6A        ROR A       Ergebnis in Bit7 setzen
9A68  28        PLP         vorheriges Ergebnis holen
9A69  29 80     AND #$80    letztes Ergebnis herstellen
9A6B  90 04     BCC $9A71   ist im ersten Test Spur0 aktiv ?
9A6D  10 1D     BPL $9A8C   nein, ist Spur0 jetzt erreicht ?
9A6F  30 02     BMI $9A73   ja, immer Sprung nach $9A73
9A71  30 19     BMI $9A8C   ist Spur0 immer noch aktiv ?
-----------------------------------------------------------------
Zustand der Spur0-Lichtschranke hat sich nicht geändert
9A73  88        DEY         ja, noch ein Versuch
9A74  D0 E8     BNE $9A5E   schon alle Abtastversuche durchgeführt ?
9A76  B0 14     BCS $9A8C   ja, war Spur0 gesetzt ?
9A78  A5 7B     LDA $7B     ja, aktuelles Kopfsteuerbyte bei Fehlern
9A7A  D0 10     BNE $9A8C   gesetzt ?
9A7C  AD 00 1C  LDA $1C00   nein, Laufwerkssteuerregister holen
9A7F  29 03     AND #$03    und Stepperbits holen
9A81  D0 09     BNE $9A8C   eine Stepperspule aktiv ?
9A83  68        PLA         nein, Laufwerksstatus wieder holen
9A84  A8        TAY         und merken
9A85  A9 00     LDA #$00    Zähler für
9A87  85 4A     STA $4A     zu fahrenden Schritte löschen
9A89  4C C9 9A  JMP $9AC9   Ende
-----------------------------------------------------------------
Zustand der Spur0-Lichtschranke hat sich geändert
9A8C  68        PLA         Laufwerksstatus wieder holen
9A8D  A8        TAY         und merken
9A8E  E6 4A     INC $4A     Zähler einen Schritt nach außen
9A90  AD 00 1C  LDA $1C00   Steuerregister holen
9A93  38        SEC         und Stepperbits für
```

```
9A94     E9 01      SBC #$01      einen Schritt nach außen
9A96     4C BB 9A   JMP $9ABB     setzen
9A99[1]  A9 02      LDA #$02      Verzögerungszähler auf zwei weitere
9A9B     85 48      STA $48       IRQs setzen
9A9D     85 62      STA $62       Stepperflag auf 'Beruhigungsphase'
9A9F     4C C9 9A   JMP $9AC9     zurück zur aufrufenden Routine
9AA2[1]  C6 48      DEC $48       Verzögerung für Kopfberuhigungszeit
9AA4     D0 23      BNE $9AC9     Kopf bereit ?
9AA6     A5 20      LDA $20       ja, Laufwerksstatus holen
9AA8     29 BF      AND #$BF      und Flag für 'Stepper an'
9AAA     85 20      STA $20       löschen
9AAC     A9 00      LDA #$00      Stepperflag
9AAE     85 62      STA $62       zurücksetzen
9AB0     4C C9 9A   JMP $9AC9     zurück zur aufrufenden Routine
----------------------------------------------------------------------
[9A58] Einen Halbspurschritt nach innen
9AB3     C6 4A      DEC $4A       Schrittzähler einen Schritt nach innen
9AB5     AD 00 1C   LDA $1C00     Steuerregister holen
9AB8     18         CLC           und Stepperbits für
9AB9     69 01      ADC #$01      einen Halbspurschritt nach innen
----------------------------------------------------------------------
[9A96] Steppersteuerung setzen
9ABB     29 03      AND #$03      setzen
9ABD     85 4B      STA $4B       Wert merken
9ABF     AD 00 1C   LDA $1C00     Steuerregister holen
9AC2     29 FC      AND #$FC      und neuen Wert
9AC4     05 4B      ORA $4B       der Stepperbits
9AC6     8D 00 1C   STA $1C00     einblenden
9AC9[6]  60         RTS           zurück zur aufrufenden Routine
[9B6C] Spur formatieren
9B89     A5 3B      LDA $3B       Befehlsnummer holen und Flags testen
9B8B     10 03      BPL $9B90     soll Spurkapazität ermittelt werden ?
9B8D     20 DC 9A   JSR $9ADC     ja, Spurkapazität feststellen
9B90[1]  AD 26 06   LDA $0626     [Fehler siehe 7.1.5]
9B93     18         CLC           [unsinnige Operation]
9B94     A9 03      LDA #$03      Zeiger $32/$33
9B96     85 33      STA $33       auf Beginn des
9B98     A9 00      LDA #$00      Datenpuffers 0
9B9A     85 32      STA $32       setzen
9B9C     8D 28 06   STA $0628     erste Sektornummer setzen (0)
9B9F     A0 00      LDY #$00      Pufferzeiger auf zurücksetzen
```

```
9BA1[1] A5 39     LDA $39         Kennzeichen für Sektorheader
9BA3  91 32       STA ($32),Y     in Puffer schreiben
9BA5  C8          INY             Pufferzeiger auf nächstes Byte setzen
9BA6  A9 00       LDA #$00        Leerbyte für Prüfsumme
9BA8  91 32       STA ($32),Y     in Puffer schreiben
9BAA  C8          INY             Pufferzeiger auf nächstes Byte setzen
9BAB  AD 28 06    LDA $0628       Sektornummer holen und
9BAE  91 32       STA ($32),Y     in Puffer schreiben
9BB0  C8          INY             Pufferzeiger auf nächstes Byte setzen
9BB1  A5 51       LDA $51         aktuelle Spurnummer
9BB3  91 32       STA ($32),Y     in Puffer schreiben
9BB5  C8          INY             Pufferzeiger auf nächstes Byte setzen
9BB6  A5 13       LDA $13         zweites Zeichen der ID holen und
9BB8  91 32       STA ($32),Y     in Puffer schreiben
9BBA  C8          INY             Pufferzeiger auf nächstes Byte setzen
9BBB  A5 12       LDA $12         erstes Zeichen der ID holen und
9BBD  91 32       STA ($32),Y     in Puffer schreiben
9BBF  C8          INY             Pufferzeiger auf nächstes Byte setzen
9BC0  A9 0F       LDA #$0F        Wert für Leerbyte
9BC2  91 32       STA ($32),Y     in Puffer schreiben
9BC4  C8          INY             Pufferzeiger auf nächstes Byte setzen
9BC5  91 32       STA ($32),Y     in Puffer schreiben
9BC7  C8          INY             Pufferzeiger auf nächstes Byte setzen
9BC8  98          TYA             Pufferzeiger holen
9BC9  48          PHA             und retten
9BCA  A2 07       LDX #$07        Zahl der einzurechnenden Bytes
9BCC  A9 00       LDA #$00        Prüfsumme
9BCE  85 3A       STA $3A         löschen
9BD0[1] 88        DEY             Pufferzeiger auf vorheriges Byte setzen
9BD1  B1 32       LDA ($32),Y     Byte aus Headerpuffer holen
9BD3  45 3A       EOR $3A         und in Prüfsumme einrechnen
9BD5  85 3A       STA $3A         Wert merken
9BD7  CA          DEX             noch ein Byte
9BD8  D0 F6       BNE $9BD0       schon ganzer Header eingerechnet ?
9BDA  91 32       STA ($32),Y     ja, Prüfsumme in Header schreiben
9BDC  68          PLA             aktuellen Pufferzeiger
9BDD  A8          TAY             wieder setzen
9BDE  EE 28 06    INC $0628       nächsten Sektor anwählen
9BE1  AD 28 06    LDA $0628       aktuelle Sektornummer
9BE4  C5 43       CMP $43         mit maximaler Nummer vergleichen
9BE6  90 B9       BCC $9BA1       ist Sektornummer erlaubt ?
```

```
9BE8     A9 03     LDA #$03       nein, Pufferzeiger für Puffer $0300
9BEA     85 31     STA $31        initialisieren
9BEC     20 30 FE  JSR $FE30      Blockheader in GCR-Bytes umwandeln
9BEF     A0 BA     LDY #$BA       Pufferzeiger auf Zusatzpuffer richten
9BF1[1]  B1 32     LDA ($32),Y    Byte aus Zusatzpuffer holen
9BF3     A2 45     LDX #$45       Zeiger auf zweiten Pufferbereich
9BF5     86 32     STX $32        setzen
9BF7     91 32     STA ($32),Y    GCR-Byte in höheren Pufferteil schreiben
9BF9     A2 00     LDX #$00       Zeiger wieder auf Anfang
9BFB     86 32     STX $32        setzen
9BFD     88        DEY            Pufferzeiger auf nächstes Byte setzen
9BFE     C0 FF     CPY #$FF       mit Endwert vergleichen
9C00     D0 EF     BNE $9BF1      Bereich $300-$344 nach $345-$389 kopiert
?
9C02     A0 44     LDY #$44       Pufferzeiger auf Zusatzpuffer richten
9C04[1]  B9 BB 01  LDA $01BB,Y    Byte aus Zusatzpuffer holen und
9C07     91 32     STA ($32),Y    in Datenpuffer schreiben
9C09     88        DEY            Pufferzeiger auf nächstes Byte setzen
9C0A     10 F8     BPL $9C04      schon ganzer Puffer übertragen ?
9C0C     18        CLC            ja, Pufferzeiger auf
9C0D     A9 03     LDA #$03       neuen Puffer für
9C0F     69 02     ADC #$02       Datenblockinhalt
9C11     85 31     STA $31        setzen
9C13     A9 00     LDA #$00       Wert für Füllbyte
9C15     A8        TAY            Pufferzeiger löschen
9C16[1]  91 30     STA ($30),Y    Leerbyte in Puffer schreiben
9C18     C8        INY            Pufferzeiger auf nächstes Byte setzen
9C19     D0 FB     BNE $9C16      schon ganzer Puffer gelöscht ?
9C1B     20 E9 F5  JSR $F5E9      ja, Prüfsumme berechnen
9C1E     85 3A     STA $3A        und merken
9C20     20 8F F7  JSR $F78F      Puffer in GCR-Bytes umwandeln
9C23     A9 00     LDA #$00       Zeiger auf aktuelle Position im
9C25     85 1B     STA $1B        Headerpuffer löschen
9C27     A2 06     LDX #$06       1536 mal $55 (%01010101)
9C29     20 63 9D  JSR $9D63      auf Diskette schreiben
9C2C[1]  A0 05     LDY #$05       Zahl der Sync-Bytes
9C2E[2]  2C 0F 18  BIT $180F      auf 'Byte Ready'-Signal
9C31     30 FB     BMI $9C2E      warten
9C33     A9 FF     LDA #$FF       Byte für Sync-Markierung
9C35     8D 01 1C  STA $1C01      auf Diskette schreiben
9C38     2C 00 1C  BIT $1C00      Sync-Flag zurücksetzen
```

```
9C3B  88        DEY          nächstes Byte
9C3C  D0 F0     BNE $9C2E    schon ganze Markierung geschrieben ?
9C3E  A2 0A     LDX #$0A     ja, Zahl der Headerbytes
9C40  A4 1B     LDY $1B      Pufferzeiger holen
9C42  2C 0F 18  BIT $180F    auf 'Byte Ready'-Signal
9C45  30 FB     BMI $9C42    warten
9C47  B1 32     LDA ($32),Y  Byte aus Headerpuffer holen
9C49  8D 01 1C  STA $1C01    und auf Diskette schreiben
9C4C  2C 00 1C  BIT $1C00    Sync-Flag zurücksetzen
9C4F  C8        INY          Pufferzeiger auf nächstes Byte setzen
9C50  CA        DEX          Zahl der Headerbytes
9C51  D0 EF     BNE $9C42    schon ganzer Header geschrieben ?
9C53  A0 09     LDY #$09     ja, Zahl der Bytes für Lücke
9C55  2C 0F 18  BIT $180F    auf 'Byte Ready'-Signal
9C58  30 FB     BMI $9C55    warten
9C5A  A9 55     LDA #$55     Leerbyte in Lücke zwischen Header und
9C5C  8D 01 1C  STA $1C01    Datenblock schreiben
9C5F  2C 00 1C  BIT $1C00    Steuerregister zurücksetzen
9C62  88        DEY          Zahl der Lückenbytes
9C63  D0 F0     BNE $9C55    Lücke geschrieben ?
9C65  A9 FF     LDA #$FF     Sync-Markierung für
9C67  A0 05     LDY #$05     Datenblockbeginn schreiben
9C69  2C 0F 18  BIT $180F    auf 'Byte Ready'-Signal
9C6C  30 FB     BMI $9C69    warten
9C6E  8D 01 1C  STA $1C01    Sync-Byte auf Diskette schreiben
9C71  2C 00 1C  BIT $1C00    Eingang für Sync Signal initialisieren
9C74  88        DEY          nächstes Byte
9C75  D0 F2     BNE $9C69    ist Sync-Narkierung geschrieben ?
9C77  A0 BB     LDY #$BB     ja, Pufferzeiger auf Zusatzpuffer setzen
9C79  2C 0F 18  BIT $180F    auf 'Byte Ready'-Signal
9C7C  30 FB     BMI $9C79    warten
9C7E  B9 00 01  LDA $0100,Y  Datenbyte holen
9C81  8D 01 1C  STA $1C01    und auf Diskette schreiben
9C84  2C 00 1C  BIT $1C00    Eingang für Byte Ready initialisieren
9C87  C8        INY          Pufferzeiger auf nächstes Byte setzen
9C88  D0 EF     BNE $9C79    ganzer Zusatzpuffer geschrieben ?
9C8A  2C 0F 18  BIT $180F    ja, auf 'Byte Ready'-Signal
9C8D  30 FB     BMI $9C8A    warten bis letztes Byte geschrieben
9C8F  B1 30     LDA ($30),Y  Byte aus Datenpuffer holen
9C91  8D 01 1C  STA $1C01    und auf Diskette schreiben
9C94  2C 00 1C  BIT $1C00    Eingang für Sync-Signal initialisieren
```

```
9C97     C8         INY           Pufferzeiger auf nächstes Byte setzen
9C98     D0 F0      BNE $9C8A     Datenpuffer auf Diskette geschrieben ?
9C9A     A9 55      LDA #$55      ja, Füllbyte für Lücke zwischen Sektoren
9C9C     AC 26 06   LDY $0626     Zahl der Bytes zwischen Sektoren
9C9F[2]  2C 0F 18   BIT $180F     auf 'Byte Ready'-Signal
9CA2     30 FB      BMI $9C9F     warten
9CA4     8D 01 1C   STA $1C01     Byte auf Diskette schreiben
9CA7     2C 00 1C   BIT $1C00     Eingang für Byte Ready initialisieren
9CAA     88         DEY           nächstes Byte
9CAB     D0 F2      BNE $9C9F     Lücke geschrieben ?
9CAD     A5 1B      LDA $1B       ja, Zeiger in Headerpuffer holen
9CAF     18         CLC           und Zahl der GCR-Bytes
9CB0     69 0A      ADC #$0A      je Header dazurechnen
9CB2     85 1B      STA $1B       neuen Zeiger merken
9CB4     CE 28 06   DEC $0628     Zahl der Sektoren erniedrigen
9CB7     F0 03      BEQ $9CBC     alle Sektoren geschrieben ?
9CB9     4C 2C 9C   JMP $9C2C     nein, nächsten Sektor schreiben
9CBC[2]  2C 0F 18   BIT $180F     auf 'Byte Ready'-Signal
9CBF     30 FB      BMI $9CBC     warten bis letztes Byte geschrieben ist
9CC1     2C 00 1C   BIT $1C00     Eingang für Byte Ready initialisieren
9CC4[1]  2C 0F 18   BIT $180F     auf 'Byte Ready'-Signal
9CC7     30 FB      BMI $9CC4     warten
9CC9     2C 00 1C   BIT $1C00     Eingang für Byte Ready initialisieren
9CCC     20 00 FE   JSR $FE00     Kopf auf Lesen umschalten
9CCF     A9 C8      LDA #$C8      Zahl der Leseversuche (200)
9CD1     8D 23 06   STA $0623     festlegen
9CD4[1]  A9 00      LDA #$00      Pufferzeiger auf aktuellen
9CD6     85 1B      STA $1B       Header löschen
9CD8     A5 43      LDA $43       Zahl der Sektoren auf Spur holen
9CDA     8D 28 06   STA $0628     und in Zähler setzen
9CDD[1]  20 54 97   JSR $9754     auf nächste Sync-Markierung warten
9CE0     A2 0A      LDX #$0A      Zahl der Headerbytes
9CE2     A4 1B      LDY $1B       Zeiger auf aktuellen Header holen
9CE4[1]  B1 32      LDA ($32),Y   und erstes Headerbyte holen
9CE6[1]  2C 0F 18   BIT $180F     auf 'Byte Ready'-Signal
9CE9     30 FB      BMI $9CE6     warten
9CEB     CD 01 1C   CMP $1C01     Byte von Diskette mit Header vergleichen
9CEE     D0 0E      BNE $9CFE     identisch ?
9CF0     C8         INY           ja, Zeiger auf nächstes Byte setzen
9CF1     CA         DEX           Zahl der Headerbytes
9CF2     D0 F0      BNE $9CE4     schon ganzer Header geprüft ?
```

```
9CF4    18          CLC           ja, Pufferzeiger
9CF5    A5 1B       LDA $1B       auf nächsten
9CF7    69 0A       ADC #$0A      Sektorheader im Puffer
9CF9    85 1B       STA $1B       richten
9CFB    4C 08 9D    JMP $9D08     Datenblock prüfen
9CFE[2] CE 23 06    DEC $0623     Zahl der Leseversuche erniedrigen
9D01    D0 D1       BNE $9CD4     noch ein Versuch ?
9D03    A9 06       LDA #$06      nein, Nummer für 'Formatierfehler'
9D05    4C 51 9D    JMP $9D51     Rückmeldung übergeben
9D08[1] 20 54 97    JSR $9754     auf nächste Sync-Markierung warten
9D0B    A0 BB       LDY #$BB      Pufferzeiger auf Zusatzpuffer richten
9D0D[1] B9 00 01    LDA $0100,Y   Byte aus Puffer holen
9D10[1] 2C 0F 18    BIT $180F     auf 'Byte Ready'-Signal
9D13    30 FB       BMI $9D10     warten
9D15    CD 01 1C    CMP $1C01     Puffer mit Diskette vergleichen
9D18    D0 E4       BNE $9CFE     identisch ?
9D1A    C8          INY           ja, Pufferzeiger auf nächstes Byte
9D1B    D0 F0       BNE $9D0D     schon ganzen Puffer überprüft ?
9D1D[1] B1 30       LDA ($30),Y   ja, Byte aus Datenpuffer holen
9D1F[1] 2C 0F 18    BIT $180F     auf 'Byte Ready'-Signal
9D22    30 FB       BMI $9D1F     warten
9D24    CD 01 1C    CMP $1C01     und Byte mit Diskette vergleichen
9D27    D0 D5       BNE $9CFE     identisch ?
9D29    C8          INY           ja, Pufferzeiger auf nächstes Byte
9D2A    D0 F1       BNE $9D1D     ganzer Puffer überprüft ?
9D2C    CE 28 06    DEC $0628     ja, nächsten Sektor
9D2F    D0 AC       BNE $9CDD     schon alle Sektoren gelesen ?
9D31    E6 51       INC $51       ja, Zeiger auf nächste Spur setzen
9D33    A5 51       LDA $51       aktuelle Formatierspur holen
9D35    2C B1 01    BIT $01B1     Flag für Diskettenseite testen
9D38    30 03       BMI $9D3D     ist 2. Seite gesetzt ?
9D3A    C9 24       CMP #$24      nein, Spur auf maximaler Spur prüfen
9D3C    2C          .byte $2C     nächste 2 Bytes überspringen (Bitbefehl)
9D3D[1] C9 47       CMP #$47      Spur mit maximaler Spur vergleichen (71)
9D3F    B0 03       BCS $9D44     ist aktuelle Spur kleiner ?
9D41    4C CA 99    JMP $99CA     ja, Stepper auf nächste Spur bewegen
9D44[1] A9 FF       LDA #$FF      Flag für aktuelle
9D46    85 51       STA $51       Formatierspur löschen
9D48    A9 00       LDA #$00      Flag für 'Pufferdaten in GCR'
9D4A    85 50       STA $50       löschen
9D4C    A9 01       LDA #$01      Nummer für 'Ok' Meldung
```

```
9D4E  4C B5 99   JMP $99B5      Rückmeldung übergeben
----------------------------------------------------------------------
[9B70/9D05]
Formatieren beenden
9D51  CE 20 06   DEC $0620      noch auszuführende Jobschleifenaufrufe
9D54  F0 03      BEQ $9D59      Stepperschleife nochmal aufrufen ?
9D56  4C CA 99   JMP $99CA      ja, Stepperkommandos ausführen
9D59  A0 FF      LDY #$FF       Flag für 'Formatieren im Gang'
9D5B  84 51      STY $51        löschen
9D5D  C8         INY            Flag für 'Pufferdaten in GCR-Code'
9D5E  84 50      STY $50        löschen
9D60  4C B5 99   JMP $99B5      Rückmeldung übergeben
----------------------------------------------------------------------
[8D56/9AE0/9C29]
X mal 256 Bytes $55 (%01010101) auf Diskette schreiben
9D63  AD 0C 1C   LDA $1C0C      Steuerregister holen
9D66  29 1F      AND #$1F       und Kopfelektronik
9D68  09 C0      ORA #$C0       auf Schreiben
9D6A  8D 0C 1C   STA $1C0C      umstellen
9D6D  A9 FF      LDA #$FF       Kopfregister
9D6F  8D 03 1C   STA $1C03      auf Ausgang schalten
9D72  A9 55      LDA #$55       Leerbyte $55
9D74  A0 00      LDY #$00       Zähler initialisieren
9D76  2C 0F 18   BIT $180F      auf 'Byte Ready'-Signal
9D79  30 FB      BMI $9D76      warten
9D7B  2C 00 1C   BIT $1C00      Eingang für 'Byte Ready' initialisieren
9D7E  8D 01 1C   STA $1C01      Byte auf Diskette schreiben
9D81  88         DEY            Zähler erniedrigen
9D82  D0 F2      BNE $9D76      schon 256 Bytes geschrieben
9D84  CA         DEX            Blockzähler erniedrigen
9D85  D0 EF      BNE $9D76      noch einmal 256 Bytes schreiben ?
9D87  60         RTS            nein, zurück zur aufrufenden Routine
----------------------------------------------------------------------
[über Vektor 02A9 von FE67/BF00]
1541 Interruptroutine für Bus- und Diskcontroller
9D88  48         PHA            Akku retten
9D89  8A         TXA            X-Register
9D8A  48         PHA            retten
9D8B  98         TYA            Y-Register
9D8C  48         PHA            retten
9D8D  AD 0D 40   LDA $400D      Flag für Interrupt durch serielles
```

```
9D90   29 08      AND #$08      Ein-/Ausgaberegister holen
9D92   F0 26      BEQ $9DBA     ist Flag gesetzt ?
9D94   2C AF 02   BIT $02AF     ja, Flag für IRQ Modus
9D97   30 21      BMI $9DBA     auf 1571 IRQ Routine umschalten ?
9D99   AD 0F 18   LDA $180F     ja, Elektronik
9D9C   09 20      ORA #$20      auf 1571 Modus
9D9E   8D 0F 18   STA $180F     schalten (2 MHz Takt)
9DA1   A9 DE      LDA #$DE      Interrupt-Vektor
9DA3   8D A9 02   STA $02A9     in $02A9/$02AA
9DA6   A9 9D      LDA #$9D      auf Routine $9DDE
9DA8   8D AA 02   STA $02AA     richten
9DAB   A9 40      LDA #$40      Timer 1 (High-Byte)
9DAD   8D 07 1C   STA $1C07     auf etwa 8 ms (2 MHz Takt)
9DB0   8D 05 1C   STA $1C05     einstellen
9DB3   A9 00      LDA #$00      Flag für Stepperphase
9DB5   85 62      STA $62       löschen
9DB7   4C EA 9D   JMP $9DEA     1571 Jobschleife
9DBA[2] AD 0D 18  LDA $180D     Interrupt Flags testen
9DBD   29 02      AND #$02      und CA1 Eingang isolieren
9DBF   F0 03      BEQ $9DC4     ist ATN aufgetreten ?
9DC1   20 53 E8   JSR $E853     ja, Flags f. Interrupt vom seriellen Bus
9DC4[1] AD 0D 1C  LDA $1C0D     Interruptflagregister holen
9DC7   0A         ASL A         und Flag für Timer 1 testen
9DC8   10 03      BPL $9DCD     Zeitgeber abgelaufen ?
9DCA   20 B0 F2   JSR $F2B0     ja, zu 1541 Kontroller-Routine
9DCD[1] BA        TSX           Stapelzeiger holen
9DCE   BD 04 01   LDA $0104,X   und Status vom Stapel holen
9DD1   29 10      AND #$10      Flag für Einprung durch 'BRK' prüfen
9DD3   F0 03      BEQ $9DD8     wurde Interrupt durch 'BRK' aufgerufen ?
9DD5   20 B0 F2   JSR $F2B0     ja, 1541 Kontroller-Routine ausführen
9DD8[1] 68        PLA           Y-Register wieder auf
9DD9   A8         TAY           Ausgangswert setzen
9DDA   68         PLA           X-Register wieder auf
9DDB   AA         TAX           Ausgangswert setzen
9DDC   68         PLA           Akku wieder holen
9DDD   40         RTI           zurück zur Unterbrechungsstelle
-----------------------------------------------------------------
[über Vektor 02A9 von FE67/BF03]
1541 Interruptroutine für Bus- und Diskcontroller
9DDE   48         PHA           Akku retten
9DDF   8A         TXA           X-Register
```

```
9DE0  48        PHA             retten
9DE1  98        TYA             Y-Register
9DE2  48        PHA             retten
9DE3  AD 0D 40  LDA $400D       Flag für Interrupt durch serielles
9DE6  29 08     AND #$08        Ein-/Ausgaberegister holen
9DE8  F0 08     BEQ $9DF2       ist Flag gesetzt ?
9DEA  A5 37     LDA $37         Busstatusbyte holen
9DEC  09 40     ORA #$40        und Flag für '1571 Bus Modus'
9DEE  85 37     STA $37         setzen
9DF0  D0 22     BNE $9E14       immer Sprung nach $9E14
9DF2  AD 0D 18  LDA $180D       Interrupt Flags testen
9DF5  29 02     AND #$02        und CA1 Eingang isolieren
9DF7  F0 07     BEQ $9E00       ist ATN aufgetreten ?
9DF9  2C 01 18  BIT $1801       ja, Flag wieder zurücksetzen
9DFC  A9 01     LDA #$01        Flag für 'ATN aufgetreten'
9DFE  85 7C     STA $7C         setzen
9E00  BA        TSX             Stapelzeiger holen
9E01  BD 04 01  LDA $0104,X     und Status vom Stapel holen
9E04  29 10     AND #$10        Flag für 'Einprung durch BRK' prüfen
9E06  F0 03     BEQ $9E0B       wurde Interrupt durch 'BRK' aufgerufen ?
9E08  20 BA 92  JSR $92BA       ja, 1571 Jobschleife ausführen
9E0B  AD 0D 1C  LDA $1C0D       Interruptflagregister holen
9E0E  0A        ASL A           und Flag für Timer 1 testen
9E0F  10 03     BPL $9E14       Zeitgeber abgelaufen ?
9E11  20 BA 92  JSR $92BA       ja, 1571 Jobschleife ausführen
9E14  68        PLA             Y-Register wieder auf
9E15  A8        TAY             Ausgangswert setzen
9E16  68        PLA             X-Register wieder auf
9E17  AA        TAX             Ausgangswert setzen
9E18  68        PLA             Akku wieder holen
9E19  40        RTI             zurück zur Unterbrechungsstelle
-----------------------------------------------------------------
9E1A  FF ...                    unbenutzter
9F0C  ... FF                    ROM-Bereich
-----------------------------------------------------------------
Tabellen für Umwandlung von 5 GCR-Bytes in 4 Binärbytes
($FF bedeutet, daß dieser GCR-Wert nicht existiert)
-----------------------------------------------------------------
```

```
[9632/9650/993B/9F0F:9683,96F1,9947/9F1D:96A0,994E/9F2A:96C4,995A]
Tabelle für GCR-Werte 2, 4, 5 und 7
9F0D 0C 04 05 FF FF 02 03 FF 0F 06 07 FF 09 0A 0B FF
9F1D 0D 0E 80 FF 00 00 10 40 FF 20 C0 60 40 A0 50 E0
9F2D FF FF FF 02 20 08 30 FF FF 00 F0 FF 60 01 70 FF
9F3D FF FF 90 03 A0 0C B0 FF FF 04 D0 FF E0 05 80 FF
9F4D 90 FF 08 0C FF 0F 09 0D 80 02 FF FF FF 03 FF FF
9F5D 00 FF FF 0F FF 0F FF FF 10 06 FF FF FF 07 00 20
9F6D A0 FF FF 06 FF 09 FF FF C0 0A FF FF FF 0B FF FF
9F7D 40 FF FF 07 FF 0D FF FF 50 0E FF FF FF FF 10 30
9F8D B0 FF 00 04 02 06 0A 0E 80 FF FF FF FF FF FF FF
9F9D 20 FF 08 09 80 10 C0 50 30 30 F0 70 90 B0 D0 FF
9FAD FF FF 00 0A FF FF FF FF F0
-----------------------------------------------------------------
[864F/866B/8697/A424/A439/A450/D651/D6D9]
Jobschleife aufrufen und Job ausführen
9FB6  00        BRK          Jobschleife aufrufen
9FB7  EA        NOP          Rücksprungadresse ausgleichen
9FB8  B5 00     LDA $00,X    Jobregister holen
9FBA  30 FC     BMI $9FB8    ist Job ausgeführt ?
9FBC  60        RTS          ja, zurück zur aufrufenden Routine
-----------------------------------------------------------------
Tabelle für GCR-Werte 2, 4, 5 und 7  (2.Teil)
9FBD 60 FF 01 0B FF FF FF FF 70 FF FF FF FF FF C0 F0
9FCD D0 FF 01 05 03 07 0B FF 90 FF FF FF FF FF FF FF
9FDD A0 FF 0C 0D FF FF FF FF B0 FF FF FF FF FF 40 60
9FED E0 FF 04 0E FF FF FF FF D0 FF FF FF FF FF FF FF
9FFD E0 FF 05 FF FF FF FF FF FF FF FF FF FF FF 50 70
-----------------------------------------------------------------
[9619/9644/9936]  Tabelle für GCR-Wert 1
A00D 0C 04 05 FF FF 02 03 FF 0F 06 07 FF 09 0A 0B FF
A01D 0D 0E 80 FF 00 00 10 40 FF 20 C0 60 40 A0 50 E0
A02D FF FF FF 02 20 08 30 30 30 00 F0 FF 60 01 70 FF
A03D FF FF 90 03 A0 0C B0 FF FF 04 D0 FF E0 05 80 FF
A04D 90 FF 08 0C FF 0F 09 0D 80 80 80 80 80 80 80 80
A05D 00 00 00 00 00 00 00 00 10 10 10 10 10 10 10 10
A06D A0 FF FF 06 FF 09 FF FF C0 C0 C0 C0 C0 C0 C0 C0
A07D 40 40 40 40 40 40 40 40 50 50 50 50 50 50 50 50
A08D B0 FF 00 04 02 06 0A 0E 80 80 80 80 80 80 80 80
A09D 20 20 20 20 20 20 20 20 30 30 30 30 30 30 30 30
A0AD FF FF 00 0A 0A 0A 0A 0A F0 F0 F0 F0 F0 F0 F0 F0
```

```
A0BD 60 60 60 60 60 60 60 60 70 70 70 70 70 70 70 70
A0CD D0 FF 01 05 03 07 0B FF 90 90 90 90 90 90 90 90
A0DD A0 A0 A0 A0 A0 A0 A0 A0 B0 B0 B0 B0 B0 B0 B0 B0
A0ED E0 FF 04 0E FF FF FF FF D0 D0 D0 D0 D0 D0 D0 D0
A0FD E0 E0 E0 E0 E0 E0 E0 E0 05 05 05 05 05 05 50 70
```

[966A/96DA/9942] Tabelle für GCR-Wert 3

```
A10D FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
A11D FF FF 80 80 00 00 10 10 FF FF C0 C0 40 40 50 50
A12D FF FF FF FF 20 20 30 30 FF FF F0 F0 60 60 70 70
A13D FF FF 90 90 A0 A0 B0 B0 FF FF D0 D0 E0 E0 FF FF
A14D FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
A15D FF FF 80 80 00 00 10 10 FF FF C0 C0 40 40 50 50
A16D FF FF FF FF 20 20 30 30 FF FF F0 F0 60 60 70 70
A17D FF FF 90 90 A0 A0 B0 B0 FF FF D0 D0 E0 E0 FF FF
A18D FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
A19D FF FF 80 80 00 00 10 10 FF FF C0 C0 40 40 50 50
A1AD FF FF FF FF 20 20 30 30 FF FF F0 F0 60 60 70 70
A1BD FF FF 90 90 A0 A0 B0 B0 FF FF D0 D0 E0 E0 FF FF
A1CD FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
A1DD FF FF 80 80 00 00 10 10 FF FF C0 C0 40 40 50 50
A1ED FF FF FF FF 20 20 30 30 FF FF F0 F0 60 60 70 70
A1FD FF FF 90 90 A0 A0 B0 B0 FF FF D0 D0 E0 E0 FF FF
```

[96A8/9953] Tabelle für GCR-Wert 6

```
A20D FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
A21D FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
A22D FF FF FF FF 08 08 08 08 00 00 00 00 01 01 01 01
A23D FF FF FF FF 0C 0C 0C 0C 04 04 04 04 05 05 05 05
A24D FF FF FF FF FF FF FF FF 02 02 02 02 03 03 03 03
A25D FF FF FF FF 0F 0F 0F 0F 06 06 06 06 07 07 07 07
A26D FF FF FF FF 09 09 09 09 0A 0A 0A 0A 0B 0B 0B 0B
A27D FF FF FF FF 0D 0D 0D 0D 0E 0E 0E 0E FF FF FF FF
A28D FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
A29D FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
A2AD FF FF FF FF 08 08 08 08 00 00 00 00 01 01 01 01
A2BD FF FF FF FF 0C 0C 0C 0C 04 04 04 04 05 05 05 05
A2CD FF FF FF FF FF FF FF FF 02 02 02 02 03 03 03 03
A2DD FF FF FF FF 0F 0F 0F 0F 06 06 06 06 07 07 07 07
A2ED FF FF FF FF 09 09 09 09 0A 0A 0A 0A 0B 0B 0B 0B
A2FD FF FF FF FF 0D 0D 0D 0D 0E 0E 0E 0E FF FF FF FF
```

```
---------------------------------------------------------------------------
[96CC/995F]  Tabelle für GCR-Wert 8
A30D FF FF FF FF FF FF FF FF FF 08 00 01 FF 0C 04 05
A31D FF FF 02 03 FF 0F 06 07 FF 09 0A 0B FF 0D 0E FF
A32D FF FF FF FF FF FF FF FF FF 08 00 01 FF 0C 04 05
A33D FF FF 02 03 FF 0F 06 07 FF 09 0A 0B FF 0D 0E FF
A34D FF FF FF FF FF FF FF FF FF 08 00 01 FF 0C 04 05
A35D FF FF 02 03 FF 0F 06 07 FF 09 0A 0B FF 0D 0E FF
A36D FF FF FF FF FF FF FF FF FF 08 00 01 FF 0C 04 05
A37D FF FF 02 03 FF 0F 06 07 FF 09 0A 0B FF 0D 0E FF
A38D FF FF FF FF FF FF FF FF FF 08 00 01 FF 0C 04 05
A39D FF FF 02 03 FF 0F 06 07 FF 09 0A 0B FF 0D 0E FF
A3AD FF FF FF FF FF FF FF FF FF 08 00 01 FF 0C 04 05
A3BD FF FF 02 03 FF 0F 06 07 FF 09 0A 0B FF 0D 0E FF
A3CD FF FF FF FF FF FF FF FF FF 08 00 01 FF 0C 04 05
A3DD FF FF 02 03 FF 0F 06 07 FF 09 0A 0B FF 0D 0E FF
A3ED FF FF FF FF FF FF FF FF FF 08 00 01 FF 0C 04 05
A3FD FF FF 02 03 FF 0F 06 07 FF 09 0A 0B FF 0D 0E FF
---------------------------------------------------------------------------
[A783/A989]
Diskette im 1571-Commodore-Format formatieren
A40D  A9 47     LDA #$47      Nummer der größten
A40F  8D AC 02  STA $02AC     zu formatierenden Spur
A412  A9 03     LDA #$03      Nummer des aktuellen Puffers
A414  20 D3 D6  JSR $D6D3     Spur und Sektor an Jobschleife
A417  A2 03     LDX #$03      Puffer 3 anwählen
A419  A9 00     LDA #$00      Flag für Diskettenseite auf
A41B  8D B2 01  STA $01B2     Seite 1 setzen
A41E  A9 F0     LDA #$F0      Jobcode für 'Formatieren'
A420  85 3B     STA $3B       merken
A422  95 00     STA $00,X     an Jobschleife übergeben
A424  20 B6 9F  JSR $9FB6     Job (Formatieren) ausführen
A427  C9 02     CMP #$02      Rückmeldung auf 'Ok' prüfen
A429  B0 45     BCS $A470     ist Job fehlerfrei verlaufen ?
A42B  A0 03     LDY #$03      Zahl der Leseversuche (4)
A42D  A9 01     LDA #$01      ja, Spurnummer (1) an
A42F  85 0C     STA $0C       Jobschleife übergeben
A431  A9 00     LDA #$00      Sektornummer (0) für
A433  85 0D     STA $0D       Jobschleife setzen
A435  A9 80     LDA #$80      Jobcode für 'Sektor lesen'
A437  95 00     STA $00,X     an Jobschleife
```

```
A439     20 B6 9F   JSR $9FB6    Sektor 1,0 testlesen
A43C     C9 02      CMP #$02     Rückmeldung auf 'Ok' prüfen
A43E     90 05      BCC $A445    ist Job fehlerfrei verlaufen
A440     88         DEY          nächster Leseversuch
A441     10 EA      BPL $A42D    schon 4 Versuche durchgeführt ?
A443     B0 2B      BCS $A470    ja, immer Sprung nach $A470
A445[1]  A9 01      LDA #$01     Flag für Diskettenseite
A447     8D B2 01   STA $01B2    auf Seite 2 setzen
A44A     A9 F0      LDA #$F0     Jobcode für 'Formatieren'
A44C     85 3B      STA $3B      setzen
A44E     95 00      STA $00,X    an Jobschleife übergeben
A450     20 B6 9F   JSR $9FB6    Job ausführen
A453     C9 02      CMP #$02     Rückmeldung mit 'Ok' vergleichen
A455     B0 19      BCS $A470    ist Job fehlerfrei verlaufen ?
A457     A0 03      LDY #$03     ja, Zahl der Leseversuche setzen
A459[1]  A9 24      LDA #$24     Spurnummer (36)
A45B     85 0C      STA $0C      an Jobschleife
A45D     A9 00      LDA #$00     Sektornummer (0)
A45F     85 0D      STA $0D      an Jobschleife
A461     A9 80      LDA #$80     Jobcode für 'Sektor lesen'
A463     95 00      STA $00,X    setzen
A465     20 B6 9F   JSR $9FB6    Sektor 36,0 testlesen
A468     C9 02      CMP #$02     Rückmeldung auf 'Ok' prüfen
A46A     B0 01      BCS $A46D    ist ein Fehler aufgetreten ?
A46C     60         RTS          nein, zurück zur aufrufenden Routine
A46D[1]  88         DEY          nächster Versuch
A46E     10 E9      BPL $A459    schon drei Versuche durchgeführt ?
A470[3]  A2 00      LDX #$00     ja, Flagwert für 'Fehler beachten'
A472     2C 98 02   BIT $0298    Fehlerstatusflag holen
A475     8E 98 02   STX $0298    und neuen Wert setzen
A478     10 01      BPL $A47B    sollte Fehler beachtet werden ?
A47A     60         RTS          nein, zurück zur aufrufenden Routine
A47B[1]  4C 0A E6   JMP $E60A    Fehlermeldung ausgeben
-----------------------------------------------------------------------
[8294/82A1/885E/BF39]
45 Taktzyklen Verzögerung
A47E     8A         TXA          X-Register retten
A47F     A2 05      LDX #$05     Verzögerungswert setzen
A481     D0 03      BNE $A486    immer Sprung nach $A486
-----------------------------------------------------------------------
```

```
[8181/8187/8298/82A8/8F0A/8F51/903A/9056/A78E/BF33]
80 Taktzyklen Verzögerung
A483    8A          TXA             X-Register retten
A484    A2 0D       LDX #$0D        Verzögerungswert setzen
A486[1] CA          DEX             Zähler erniedrigen
A487    D0 FD       BNE $A486       Verzögerung zu Ende ?
A489    AA          TAX             ja, X-Register wieder herstellen
A48A    60          RTS             zurück zur aufrufenden Routine
-----------------------------------------------------------------
[A4C2/A508/A51E/A54F/A5A7/A678/A962]
BAM-Pufferzeiger retten
A48B    A5 6D       LDA $6D         Low-Byte holen und
A48D    8D AD 02    STA $02AD       zwischenspeichern
A490    A5 6E       LDA $6E         High-Byte holen und
A492    8D AE 02    STA $02AE       zwischenspeichern
A495    60          RTS             zurück zur aufrufenden Routine
-----------------------------------------------------------------
[A4D1/A51B/A531/A58A/A5C2/A6C7/A6E2/A97B]
BAM-Pufferzeiger wieder herstellen
A496    AD AD 02    LDA $02AD       Low-Byte aus Zwischenspeicher
A499    85 6D       STA $6D         holen und wieder setzen
A49B    AD AE 02    LDA $02AE       High-Byte aus Zwischenspeicher
A49E    85 6E       STA $6E         holen und wieder setzen
A4A0    60          RTS             zurück zur aufrufenden Routine
-----------------------------------------------------------------
[A851/A887/A8BC/A8F5/A918/A931]
Zeiger auf BAM-Muster des Sektors setzen (für Seite 2)
A4A1    A6 7F       LDX $7F         Nummer des aktuellen Laufwerks (0)
A4A3    BD FF 00    LDA $00FF,X     Laufwerksstatus holen
A4A6    F0 05       BEQ $A4AD       ist Laufwerk bereit ?
A4A8    A9 74       LDA #$74        nein, Fehlermeldung
A4AA    20 48 E6    JSR $E648       '74 drive not ready' ausgeben
A4AD[1] 20 19 F1    JSR $F119       Kanalnummer der BAM feststellen
A4B0    20 DF F0    JSR $F0DF       BAM von Diskette einlesen
A4B3    AD F9 02    LDA $02F9       Flag für 'BAM gültig/ungültig' testen
A4B6    F0 07       BEQ $A4BF       ist BAM auf Disk gültig ?
A4B8    09 80       ORA #$80        ja, Flag für 'BAM schreiben'
A4BA    8D F9 02    STA $02F9       setzen
A4BD    D0 03       BNE $A4C2       immer Sprung nach $A4C2
A4BF[1] 20 8D A5    JSR $A58D       BAM auf Diskette schreiben
A4C2[1] 20 8B A4    JSR $A48B       BAM Pufferzeiger retten
```

```
A4C5  20 34 A5   JSR $A534      neuen Pufferzeiger setzen
A4C8  A5 80      LDA $80        Nummer der aktuellen Spur holen
A4CA  38         SEC            und auf Spur
A4CB  E9 24      SBC #$24       von Seite 1 umrechnen
A4CD  A8         TAY            dann dazugehörige Zahl der freien Blocks
A4CE  B1 6D      LDA ($6D),Y    der Spur aus Puffer holen
A4D0  48         PHA            Wert merken
A4D1  20 96 A4   JSR $A496      alten Pufferzeiger wieder herstellen
A4D4  68         PLA            Zahl der Blocks wieder holen
A4D5  60         RTS            zurück zur aufrufenden Routine
--------------------------------------------------------------------
A4D6  FF ...                    unbenutzter
A4E6  ... FF                    ROM-Bereich
--------------------------------------------------------------------
[A854/A88A/A8CF]
BAM-Bit eines Sektors holen (für Seite 2)
A4E7  A5 80      LDA $80        Nummer der gewünschten Spur
A4E9  38         SEC            physikalische Spurnummer
A4EA  E9 24      SBC #$24       errechnen
A4EC  A8         TAY            und merken
A4ED  A5 81      LDA $81        Nummer des gewünschten Sektors holen
A4EF  4A         LSR A          und durch acht teilen (da 8 Bits
A4F0  4A         LSR A          pro Byte) und so entsprechdes Byte der
A4F1  4A         LSR A          drei BAM-Bytes anwählen
A4F2  18         CLC            zu Position in BAM-Muster
A4F3  79 DB A5   ADC $A5DB,Y    Position der Spur addieren
A4F6  A8         TAY            und als Zeiger auf BAM-Muster merken
A4F7  A5 81      LDA $81        Nummer des gewünschten Sektors holen
A4F9  29 07      AND #$07       und Position des BAM-Bits im
A4FB  AA         TAX            Byte-Muster bestimmen
A4FC  B9 46 01   LDA $0146,Y    Byte-Muster aus BAM-Puffer holen
A4FF  3D E9 EF   AND $EFE9,X    und Bit des Sektors isolieren
A502  08         PHP            Wert merken
A503  B9 46 01   LDA $0146,Y    gesamtes Byte-Muster nochmal holen
A506  28         PLP            und vorherigen Wert dazu holen
A507  60         RTS            zurück zur aufrufenden Routine
--------------------------------------------------------------------
[A862]
Zahl der freien Blocks einer Spur in der BAM erhöhen
A508  20 8B A4   JSR $A48B      aktuellen BAM-Zeiger retten
A50B  20 34 A5   JSR $A534      Zeiger auf BAM-Muster setzen
```

```
A50E  A5 80     LDA $80        Nummer der gewünschten Spur holen
A510  38        SEC            und physikalische Spurnummer
A511  E9 24     SBC #$24       (Seite-1-Wert) errechnen
A513  A8        TAY            und merken
A514  18        CLC            dann
A515  B1 6D     LDA ($6D),Y    Byte für Zahl der freien Blocks der
A517  69 01     ADC #$01       Spur holen und
A519  91 6D     STA ($6D),Y    um eins erhöhen
A51B  4C 96 A4  JMP $A496      aktuellen BAM-Zeiger wieder holen
-----------------------------------------------------------------
[A898]
Zahl der freien Blocks einer Spur in der BAM erniedrigen
A51E  20 8B A4  JSR $A48B      aktuellen BAM-Zeiger retten
A521  20 34 A5  JSR $A534      BAM-Zeiger auf Byte-Muster der Spur
A524  A5 80     LDA $80        Nummer der gewünschten Spur
A526  38        SEC            holen und physikalische Spurnummer
A527  E9 24     SBC #$24       (Seite-1-Wert) errechnen
A529  A8        TAY            und merken
A52A  38        SEC            Zahl der freien
A52B  B1 6D     LDA ($6D),Y    Blocks der Spur
A52D  E9 01     SBC #$01       in zugehörigem Byte
A52F  91 6D     STA ($6D),Y    der BAM erniedrigen
A531  4C 96 A4  JMP $A496      alten BAM-Zeiger wieder setzen
-----------------------------------------------------------------
[A4C5/A50B/A521/A552/A965]
Pufferzeiger für 2. Puffer vom internen Kanal 6 setzen
A534  A2 0D     LDX #$0D       Kanalnummer (6) für 2. Puffer
A536  B5 A7     LDA $A7,X      zugeordnete Puffernummer
A538  29 0F     AND #$0F       holen und
A53A  AA        TAX            merken
A53B  BD E0 FE  LDA $FEE0,X    High-Byte der Pufferadresse holen
A53E  85 6E     STA $6E        und in Pufferzeiger Übernehmen
A540  A9 DD     LDA #$DD       Low-Byte für BAM-Zeiger
A542  85 6D     STA $6D        setzen
A544  60        RTS            zurück zur aufrufenden Routine
-----------------------------------------------------------------
[A8BF/A94E]
Zahl der freien Blocks in BAM überprüfen (Seite 2 BAM)
A545  A5 6F     LDA $6F        Zwischenspeicher
A547  48        PHA            retten
A548  A5 80     LDA $80        aktuelle Spurnumer (Seite 2) holen
```

```
A54A  38        SEC            und auf physikalische Nummer
A54B  E9 24     SBC #$24       umrechnen
A54D  A8        TAY            Spurnummer
A54E  48        PHA            merken
A54F  20 8B A4  JSR $A48B      aktuellen BAM-Zeiger retten
A552  20 34 A5  JSR $A534      Zeiger auf BAM-Muster setzen
A555  B1 6D     LDA ($6D),Y    Zahl der angegebenen freien Blocks
A557  48        PHA            holen und merken
A558  A9 00     LDA #$00       Zwischenspeicher für Zahl der freien
A55A  85 6F     STA $6F        Blocks löschen
A55C  A9 01     LDA #$01       Zeiger auf Puffer für
A55E  85 6E     STA $6E        Rückseite setzen
A560  B9 DB A5  LDA $A5DB,Y    Position des BAM-Musters in Puffer
A563  18        CLC            holen und auf Pufferbereich
A564  69 46     ADC #$46       $0146-$01BB umrechen
A566  85 6D     STA $6D        BAM-Zeiger setzen
A568  A0 02     LDY #$02       Zahl der BAM-Muster-Bytes -1
A56A  A2 07     LDX #$07       Zahl der Bits pro Byte -1
A56C  B1 6D     LDA ($6D),Y    Bit-Muster der Sektorenbelegung holen
A56E  3D E9 EF  AND $EFE9,X    und Bit eines Sektors isolieren
A571  F0 02     BEQ $A575      ist Sektor frei ?
A573  E6 6F     INC $6F        ja, Zahl der freien Blocks erhöhen
A575  CA        DEX            nächstes Bit prüfen
A576  10 F4     BPL $A56C      ist schon ganzes Byte beachtet ?
A578  88        DEY            nächstes BAM-Byte der Spur einrechen
A579  10 EF     BPL $A56A      Blocks der gesamten Spur geprüft ?
A57B  68        PLA            ja, Zahl der in BAM angegebenen Blocks
A57C  C5 6F     CMP $6F        holen und mit neuer Zahl vergleichen
A57E  F0 05     BEQ $A585      sind Blockangaben richtig ?
A580  A9 71     LDA #$71       nein, Fehlermeldung
A582  20 45 E6  JSR $E645      '71 Dir Error' ausgeben
A585  68        PLA            Nummer der bearbeiteten Spur
A586  A8        TAY            wieder holen
A587  68        PLA            Zwischenspeicher wieder
A588  85 6F     STA $6F        herstellen
A58A  4C 96 A4  JMP $A496      BAM-Zeiger wieder herstellen
-------------------------------------------------------------------
[A4BF/EF37/F001/F09C]
1571/1541 BAM auf Diskette schreiben
A58D  AD 0F 18  LDA $180F      Steuerregister holen
A590  29 20     AND #$20       und Flag für Betriebsmodus holen
```

```
A592    D0 03      BNE $A597     ist Floppy im 1541-Modus ?
A594[1] 4C 8A D5   JMP $D58A     ja, Puffer auf Diskette schreiben (1541)
A597[1] AD AC 02   LDA $02AC     Nummer der größten Spur der Diskette
A59A    C9 25      CMP #$25      holen und auf 35 testen
A59C    90 F6      BCC $A594     hat die Diskette zwei Seiten ?
A59E    A6 F9      LDX $F9       ja, Nummer des aktuellen Puffers holen
A5A0    BD 5B 02   LDA $025B,X   dazugehörigen Jobcode
A5A3    48         PHA           feststellen und retten
A5A4    20 8A D5   JSR $D58A     Sektor auf Diskette schreiben
A5A7    20 8B A4   JSR $A48B     aktuellen BAM-Zeiger retten
A5AA    20 3A EF   JSR $EF3A     BAM in Puffer einlesen
A5AD    20 08 F0   JSR $F008     Puffer für BAM löschen
A5B0    A5 F9      LDA $F9       Nummer des aktuellen Puffers holen
A5B2    0A         ASL A         und verdoppeln (da Zeiger
A5B3    AA         TAX           auf 2-Byte-Tabelle)
A5B4    A9 35      LDA #$35      Spur 18, Seite 2 (Spur 53) an
A5B6    95 06      STA $06,X     Jobschleife übergeben
A5B8    A0 68      LDY #$68      Pufferzeiger auf Ende der 1571 BAM
A5BA[1] B9 46 01   LDA $0146,Y   Byte aus BAM-Puffer holen
A5BD    91 6D      STA ($6D),Y   und in aktuellen Datenpuffer schreiben
A5BF    88         DEY           Zeiger auf nächstes Byte richten
A5C0    10 F8      BPL $A5BA     schon ganzer Puffer kopiert ?
A5C2    20 96 A4   JSR $A496     ja, BAM-Zeiger wieder herstellen
A5C5    20 8A D5   JSR $D58A     Sektor auf Diskette schreiben
A5C8    A5 F9      LDA $F9       Nummer des aktuellen Puffers holen
A5CA    0A         ASL A         und verdoppeln (da Zeiger auf
A5CB    AA         TAX           Tabelle mit 2-Byte-Werten)
A5CC    AD 85 FE   LDA $FE85     Nummer der Directoryspur (Seite 1) holen
A5CF    95 06      STA $06,X     und an Jobschleife übergeben
A5D1    20 86 D5   JSR $D586     Sektor von Diskette einlesen
A5D4    68         PLA           Jobcode wieder holen
A5D5    A6 F9      LDX $F9       Nummer des aktuellen Puffers
A5D7    9D 5B 02   STA $025B,X   Jobcode wieder in Tabelle schreiben
A5DA    60         RTS           zurück zur aufrufenden Routine
-----------------------------------------------------------------------
[verwendet in : A4F3/A560]
Position der BAM-Muster der Spuren im BAM-Puffer
A5DB  00 03 06 09 0C 0F 12 15 18 1B 1E 21 24 27 2A 2D
A5EB  30 33 36 39 3C 3F 42 45 48 4B 4E 51 54 57 5A 5D
A5FB  60 63 66
-----------------------------------------------------------------------
```

```
[E7A3]
Routine für &-Befehl
A5FE  AD 0F 18   LDA $180F     Steuerregister holen
A601  29 20      AND #$20      und Betriebsmodus testen
A603  F0 0F      BEQ $A614     ist Floppy im 1571 Modus ?
A605  A0 00      LDY #$00      ja, [Fehler siehe 7.1.5]
A607  A2 00      LDX #$00      [unnötige Initialisierung]
A609  A9 01      LDA #$01      Zeiger auf Beginn des Eingabepuffers
A60B  8D 7A 02   STA $027A     richten
A60E  20 12 C3   JSR $C312     Laufwerksnummern holen
A611  4C A8 E7   JMP $E7A8     zurück zur 1541 &-Routine
A614  A9 8D      LDA #$8D      Endekennzeichen (Shift + Return)
A616  20 68 C2   JSR $C268     in Befehlstring suchen
A619  4C A8 E7   JMP $E7A8     zurück zur 1541 &-Routine
--------------------------------------------------------------------
[EBFC]
Befehl vom Rechner ausführen
A61C  20 46 C1   JSR $C146     Befehlsstring ausführen
A61F  20 B2 81   JSR $81B2     1571 Bus auf Eingang schalten
A622  A5 37      LDA $37       Busstatusbyte holen
A624  29 7F      AND #$7F      und Flag für '1571 Modus'
A626  85 37      STA $37       löschen
A628  4C FF EB   JMP $EBFF     zurück zur Warteschleife
--------------------------------------------------------------------
[F997]
Zähler für 'Motor ausschalten' initialisieren
A62B  A9 FF      LDA #$FF      Zähler für Nachlaufzeit des Motors
A62D  85 48      STA $48       festlegen
A62F  A9 06      LDA #$06      Zahl der noch auszuführenden
A631  85 35      STA $35       Stepperroutinen-Aufrufe
A633  60         RTS           zurück zur aufrufenden Routine
--------------------------------------------------------------------
[F9AB]
Motor anschalten und warten bis auf Drehzahl
A634  D0 07      BNE $A63D     hat sich 'Write Protect' geändert ?
A636  AD AB 02   LDA $02AB     Zähler für Hochlaufphase holen
A639  D0 10      BNE $A64B     ist Motor auf Drehzahl ?
A63B  F0 1A      BEQ $A657     ja, immer Sprung nach $A657
A63D  A9 FF      LDA #$FF      Zähler für Nachlaufzeit
A63F  8D AB 02   STA $02AB     setzen
A642  20 64 87   JSR $8764     Motor anschalten
```

```
A645    A9 01      LDA #$01        Flag für 'Diskette initialisieren'
A647    85 1C      STA $1C         setzen
A649    D0 0C      BNE $A657       immer Sprung nach $A657
A64B[1] CE AB 02   DEC $02AB       Zahl der Warte-IRQs erniedrigen
A64E    D0 07      BNE $A657       ist Motor auf Drehzahl ?
A650    A5 20      LDA $20         ja, Laufwerksstatus holen
A652    D0 03      BNE $A657       wurde Motor angeschaltet ?
A654    20 70 87   JSR $8770       nein, Motor anschalten
A657[4] 4C B1 F9   JMP $F9B1       zurück zur Kopfsteuerroutine
-----------------------------------------------------------------------
[FF15]
Ein-/Ausgaberegister initialisieren
A65A    A9 02      LDA #$02        Data-Ausgang auf
A65C    8D 00 18   STA $1800       High setzen
A65F    A9 20      LDA #$20        auf 1571 Modus schalten, Bus auf Eingang
A661    8D 01 18   STA $1801       und Kopf auf Seite 1 ansteuern
A664    4C 18 FF   JMP $FF18       zurück zur Reset-Routine
-----------------------------------------------------------------------
[D05D/F107]
1571/1541 BAM von Diskette lesen
A667    AD 0F 18   LDA $180F       Steuerregister holen
A66A    29 20      AND #$20        und auf Betriebsmodus testen
A66C    D0 03      BNE $A671       ist Floppy im 1541 Modus ?
A66E[1] 4C 86 D5   JMP $D586       ja, Sektor lesen
A671[1] AD AC 02   LDA $02AC       Nummer der größten Spur holen
A674    C9 25      CMP #$25        und mit 35 vergleichen
A676    90 F6      BCC $A66E       sind 2 Seiten verwendet ?
A678    20 8B A4   JSR $A48B       ja, aktuellen BAM-Zeiger retten
A67B    A9 00      LDA #$00        BAM-Zeiger auf Pufferbeginn
A67D    85 6D      STA $6D         richten
A67F    A6 F9      LDX $F9         Puffernummer holen
A681    BD E0 FE   LDA $FEE0,X     und High-Byte der Pufferadresse holen
A684    85 6E      STA $6E         und in Pufferzeiger setzen
A686    A9 FF      LDA #$FF        Flag für 'Fehler bei Jobausführung
A688    8D 98 02   STA $0298       nicht beachten' setzen
A68B    A5 F9      LDA $F9         Puffernummer wieder holen
A68D    0A         ASL A           und verdoppeln (da Tabelle aus je
A68E    AA         TAX             2 Parametern besteht)
A68F    A9 35      LDA #$35        Spur 18, Seite 2 (Directoryspur) an
A691    95 06      STA $06,X       Jobschleife übergeben
A693    20 86 D5   JSR $D586       Sektor lesen
```

```
A696  C9 02     CMP #$02       Rückmeldung auf Fehler testen
A698  6A        ROR A          Ergebnis in Bit7 merken (1=Fehler)
A699  29 80     AND #$80       und Bit isolieren
A69B  49 80     EOR #$80       Bit für Prüfung in $A6D5 vorbereiten
A69D  8D AF 01  STA $01AF      und merken (0= Fehler aufgetreten)
A6A0  10 0A     BPL $A6AC      ist ein Fehler aufgetreten ?
A6A2  A0 68     LDY #$68       nein, Zeiger auf Ende der 1571-BAM
A6A4  B1 6D     LDA ($6D),Y    Byte aus Datenpuffer lesen
A6A6  99 46 01  STA $0146,Y    und Byte in 1571 BAM-Puffer schreiben
A6A9  88        DEY            Zeiger auf nächstes Byte richten
A6AA  10 F8     BPL $A6A4      schon alle Bytes übertragen ?
A6AC  A9 FF     LDA #$FF       Flag für 'Fehler bei Jobausführung
A6AE  8D 98 02  STA $0298      nicht beachten' setzen
A6B1  A5 F9     LDA $F9        Nummer des aktuellen Datenpuffers
A6B3  0A        ASL A          holen und verdoppeln (da Zeiger in
A6B4  AA        TAX            2-Byte-Werte Tabelle)
A6B5  AD 85 FE  LDA $FE85      Nummer der Directoryspur (18) holen und
A6B8  95 06     STA $06,X      als Spurnummer des Jobs setzen
A6BA  20 86 D5  JSR $D586      Sektor lesen
A6BD  C9 02     CMP #$02       Rückmeldung mit 'Ok' vergleichen
A6BF  90 10     BCC $A6D1      ist Job fehlerfrei verlaufen ?
A6C1  AA        TAX            ja, Rückmeldung merken (0/1)
A6C2  A9 24     LDA #$24       Spur 35 als größte Spur setzen
A6C4  8D AC 02  STA $02AC      (d.h. nur eine Seite wird benutzt)
A6C7  20 96 A4  JSR $A496      aktuellen BAM-Zeiger wieder herstellen
A6CA  8A        TXA            Rückmeldung wieder holen
A6CB  20 0A E6  JSR $E60A      Fehlermeldung ausgeben
A6CE  4C 44 D6  JMP $D644      Job Fehlerbehandlung
A6D1  A0 03     LDY #$03       Pufferzeiger setzen und Kennzeichen
A6D3  B1 6D     LDA ($6D),Y    für 1571 Diskette ($80) holen
A6D5  2D AF 01  AND $01AF      Kennzeichen mit vorherigem Fehler prüfen
A6D8  30 03     BMI $A6DD      vorheriges Lesen und Kennzeichen ok ?
A6DA  A9 24     LDA #$24       nein, Spurzahl für eine Seite benutzt
A6DC  2C        .byte $2C      nächste 2 Bytes überspringen (Bitbefehl)
A6DD  A9 47     LDA #$47       Spurzahl für 2 Diskettenseiten
A6DF  8D AC 02  STA $02AC      setzen
A6E2  4C 96 A4  JMP $A496      BAM-Pufferzeiger wieder herstellen
--------------------------------------------------------------------------------
```

[D03F]

auf 1571 Diskette prüfen (Initialisieren)

```
A6E5  20 8C D5  JSR $D58C     Jobcode übergeben und ausführen
A6E8  48        PHA           Rückmeldung merken
A6E9  C9 02     CMP #$02      Meldung auf 'Ok' prüfen
A6EB  B0 49     BCS $A736     ist Job fehlerfrei verlaufen ?
A6ED  AD 0F 18  LDA $180F     ja, Steuerregister holen
A6F0  29 20     AND #$20      und Betriebsmodus feststellen
A6F2  F0 42     BEQ $A736     ist Floppy im 1571 Modus ?
A6F4  A9 47     LDA #$47      ja, max. Spurnummer +1 (71)
A6F6  8D AC 02  STA $02AC     setzen
A6F9  A9 FF     LDA #$FF      Flag für 'Fehler bei Jobausführung
A6FB  8D 98 02  STA $0298     nicht beachten' setzen
A6FE  A5 16     LDA $16       erstes Zeichen der ID des letzten
A700  48        PHA           Sektorheaders retten
A701  A5 17     LDA $17       zweites Zeichen der ID des letzten
A703  48        PHA           Sektorheaders retten
A704  A5 F9     LDA $F9       Nummer des aktuellen Datenpuffers holen
A706  0A        ASL A         und verdoppeln (da Tabelle aus
A707  AA        TAX           2-Byte-Werten besteht)
A708  A9 35     LDA #$35      Spur 18, Seite 2 (Directory Rückseite)
A70A  95 06     STA $06,X     an Jobschleife übergeben
A70C  A9 B0     LDA #$B0      Jobcode für 'Sektor suchen'
A70E  20 8C D5  JSR $D58C     an Jobschleife übergeben und ausführen
A711  C9 02     CMP #$02      Rückmeldung mit 'Ok' vergleichen
A713  68        PLA           zweites Zeichen der zuletzt gelesenen
A714  A8        TAY           ID wieder holen
A715  68        PLA           erstes Zeichen der zuletzt gelesenen
A716  AA        TAX           ID wieder holen
A717  B0 0B     BCS $A724     ist Job fehlerfrei verlaufen ?
A719  E4 16     CPX $16       ja, letzte ID mit neuer ID vergleichen
A71B  D0 07     BNE $A724     identisch ?
A71D  C4 17     CPY $17       ja, auch zweites ID-Zeichen vergleichen
A71F  D0 03     BNE $A724     identisch ?
A721  A9 47     LDA #$47      ja, Zahl der Spuren +1 für 2 Seiten (71)
A723  2C        .byte $2C     nächste 2 Bytes überspringen (Bitbefehl)
A724  A9 24     LDA #$24      Zahl der Spuren +1 für 1 Seite (35)
A726  8D AC 02  STA $02AC     als max. Spurnummer setzen
A729  84 17     STY $17       zuerst gelesene ID
A72B  86 16     STX $16       wieder zurücksetzen
A72D  A5 F9     LDA $F9       Nummer des aktuellen Datenpuffers holen
```

```
A72F    0A          ASL A        und verdoppeln (da Zeiger auf
A730    AA          TAX          2-Byte-Werte)
A731    AD 85 FE    LDA $FE85    Nummer der Directoryspur (18) holen
A734    95 06       STA $06,X    als Spurnummer des Jobs setzen
A736[2] 68          PLA          Wert für Rückmeldung 'Ok'
A737    60          RTS          zurück zur aufrufenden Routine
----------------------------------------------------------------------

[F005]
1571 BAM-Puffer löschen
A738    20 3A EF    JSR $EF3A    Pufferzeiger setzen
A73B    AD 0F 18    LDA $180F    Steuerregister holen
A73E    29 20       AND #$20     und Betriebsmodus testen
A740    F0 0A       BEQ $A74C    ist Floppy im 1571 Modus ?
A742    A9 00       LDA #$00     ja, Wert für Leerbyte
A744    A0 68       LDY #$68     Pufferzeiger setzen
A746[1] 99 46 01    STA $0146,Y  Byte in BAM-Puffer löschen
A749    88          DEY          Pufferzeiger auf nächstes Byte setzen
A74A    10 FA       BPL $A746    schon ganzer Puffer gelöscht ?
A74C[1] 4C 08 F0    JMP $F008    ja, Zeiger für 1541 BAM setzen
----------------------------------------------------------------------

[F24B]
physikalische (absolute) Spurnummer berechnen
A74F    48          PHA          Spurnummer merken
A750    AD 0F 18    LDA $180F    Steuerregister holen
A753    29 20       AND #$20     und Betriebsmodus testen
A755    F0 08       BEQ $A75F    ist Floppy im 1571 Modus ?
A757    68          PLA          ja, Spurnummer wieder holen und mit
A758    C9 24       CMP #$24     max. Nummer +1 (für Seite 1) vergleichen
A75A    90 04       BCC $A760    ist Spur auf Seite 2 ?
A75C    E9 23       SBC #$23     ja, Spurnummer auf Seite 1 umrechnen
A75E    24 68       .byte $24    nächstes Byte überspringen (Bit-Befehl)
A75F[1] 68          PLA          Stack wieder korregieren
A760[1] AE D6 FE    LDX $FED6    Zahl der Spurzonen holen
A763    60          RTS          zurück zur aufrufenden Routine
----------------------------------------------------------------------

[EE56]
neue BAM erzeugen (1571/1541)
A764    20 05 F0    JSR $F005    BAM-Puffer löschen
A767    AD 0F 18    LDA $180F    Steuerregister holen
A76A    29 20       AND #$20     und Betriebsmodus testen
A76C    D0 03       BNE $A771    ist Floppy im 1571 Modus ?
```

```
A76E  A9 24     LDA #$24      ja, max. Spurnummer +1 (36)
A770  2C        .byte $2C     nächste 2 Bytes überspringen (Bitbefehl)
A771  A9 47     LDA #$47      max. Spur für 2 Seiten (71)
A773  8D AC 02  STA $02AC     festlegen
A776  4C 43 EE  JMP $EE43     neue BAM erzeugen
-----------------------------------------------------------------------
[FF42]
Commodore Diskette formatieren
A779  AD 0F 18  LDA $180F     Steuerregister holen
A77C  29 20     AND #$20      und Betriebsmodus testen
A77E  D0 03     BNE $A783     ist Floppy im 1541 Modus ?
A780  4C C6 C8  JMP $C8C6     ja, 1541 Diskette formatieren
A783  4C 0D A4  JMP $A40D     1571 Diskette formatieren
-----------------------------------------------------------------------
[EBE4]
CIA 6526 bei Reset initialisieren
A786  AD 01 18  LDA $1801     Steuerregister holen
A789  29 DF     AND #$DF      und auf 1541 Betrieb
A78B  8D 01 18  STA $1801     umschalten
A78E  20 83 A4  JSR $A483     ca. 80 Taktzyklen Verzögerung
A791  A9 7F     LDA #$7F      Interruptregister
A793  8D 0D 40  STA $400D     zurücksetzen
A796  A9 08     LDA #$08      Timer A auf 'one shot' (nur ein
A798  8D 0E 40  STA $400E     Durchlauf) setzen
A79B  8D 0F 40  STA $400F     Timer B auf gleichen Modus setzen
A79E  A9 00     LDA #$00      Timer A High-Byte
A7A0  8D 05 40  STA $4005     löschen
A7A3  A9 06     LDA #$06      Timer A Low-Byte auf 6 Taktzyklen
A7A5  8D 04 40  STA $4004     setzen
A7A8  A9 01     LDA #$01      Timer A
A7AA  8D 0E 40  STA $400E     starten
A7AD  20 B2 81  JSR $81B2     1571 Bus auf Eingang
A7B0  4C 59 EA  JMP $EA59     auf ATN-Kommandomodus prüfen
-----------------------------------------------------------------------
[924B/EA68/EC04]
ATN-Kommando vom Bus annehmen
A7B3  AD 0F 18  LDA $180F     Steuerregister holen
A7B6  29 20     AND #$20      und Betriebsmodus testen
A7B8  F0 03     BEQ $A7BD     ist Floppy im 1571 Modus ?
A7BA  4C CE 80  JMP $80CE     ja, ATN-Kommando vom 1571 Bus holen
A7BD  4C 5B E8  JMP $E85B     ATN-Kommando vom 1541 Bus holen
```

```
---------------------------------------------------------------------
[EB22]
Patch (=Korrektur) für Reset-Routine
A7C0  78         SEI            Bus- und Kontrolleraufruf abschalten
A7C1  A2 45      LDX #$45       Stackpointer auf Bereich $100-$145
A7C3  9A         TXS            initialisieren
A7C4  4C 25 EB   JMP $EB25      zurück zur Reset Routine
---------------------------------------------------------------------
[ED8F/EE56]
neue 1571/1541-BAM erzeugen
A7C7  AD 0F 18   LDA $180F      Steuerregister holen
A7CA  29 20      AND #$20       und Betriebsmodus testen
A7CC  D0 09      BNE $A7D7      ist Floppy im 1541 Modus ?
A7CE  A0 03      LDY #$03       Pufferzeiger auf Kennzeichen für Disktyp
A7D0  A9 00      LDA #$00       Kennzeichen für 1541 Diskette in
A7D2  91 6D      STA ($6D),Y    BAM schreiben
A7D4  4C B7 EE   JMP $EEB7      1541 BAM erzeugen
A7D7  AD AC 02   LDA $02AC      Nummer der größten Spur holen
A7DA  C9 25      CMP #$25       und mit 37 vergleichen
A7DC  90 F0      BCC $A7CE      ist Seite 2 benutzt ?
A7DE  A0 01      LDY #$01       ja, Nummer der ersten Spur festlegen
A7E0  A2 00      LDX #$00       ersten Sektor setzen
A7E2  C0 12      CPY #$12       mit Nummer der Directoryspur vergleichen
A7E4  F0 34      BEQ $A81A      Directoryspur schon erreicht ?
A7E6  8A         TXA            nein, Nummer des aktuellen Sektors
A7E7  48         PHA            merken
A7E8  A9 00      LDA #$00       Rechenregister
A7EA  85 6F      STA $6F        für Bitmuster
A7EC  85 70      STA $70        der belegten Sektoren
A7EE  85 71      STA $71        löschen
A7F0  B9 2B 94   LDA $942B,Y    Zahl der Sektoren der Spur feststellen
A7F3  AA         TAX            und merken
A7F4  38         SEC            Flagwert für 'Sektor frei'
A7F5  26 6F      ROL $6F        in Rechenregister
A7F7  26 70      ROL $70        für Bitmuster
A7F9  26 71      ROL $71        einschieben
A7FB  CA         DEX            nächsten Sektor anwählen
A7FC  D0 F6      BNE $A7F4      schon alle Sektoren angelegt ?
A7FE  68         PLA            Nummer des ersten Sektors und
A7FF  AA         TAX            Pufferzeiger wieder setzen
A800  A5 6F      LDA $6F        erstes Byte des Bitmusters holen und
```

```
A802    9D 46 01   STA $0146,X    in BAM-Puffer schreiben
A805    A5 70      LDA $70        zweites Byte des Bitmusters holen und
A807    9D 47 01   STA $0147,X    in BAM-Puffer schreiben
A80A    A5 71      LDA $71        drittes Byte des Bitmusters holen und
A80C    9D 48 01   STA $0148,X    in BAM-Puffer schreiben
A80F    E8         INX            3 Bytes des Bitmusters
A810    E8         INX            mit Pufferzeiger
A811    E8         INX            überspringen
A812    E0 33      CPX #$33       auf Position für Directoryspur prüfen
A814    D0 04      BNE $A81A      Spur 18 schon erreicht ?
A816    E8         INX            BAM-Eintrag
A817    E8         INX            von Spur 18
A818    E8         INX            mit Pufferzeiger überspringen
A819    C8         INY            Zeiger der aktuellen Spur auf
A81A[2] C8         INY            Spur 19 setzen
A81B    C0 24      CPY #$24       mit Ende der ersten Seite vergleichen
A81D    90 C3      BCC $A7E2      ist Spur kleiner ?
A81F    20 B7 EE   JSR $EEB7      nein, 1541 BAM anlegen
A822    A0 03      LDY #$03       Pufferzeiger initialisieren
A824    A9 80      LDA #$80       Kennzeichen für 1571-Diskette
A826    91 6D      STA ($6D),Y    in Directorysektor schreiben
A828    A0 FF      LDY #$FF       Pufferzeiger setzen
A82A    A2 22      LDX #$22       Zahl der Spuren (ohne Directoryspur)
A82C[1] BD 2C 94   LDA $942C,X    Anzahl der freien Blocks der Spur
A82F    91 6D      STA ($6D),Y    in BAM-Puffer schreiben
A831    88         DEY            Pufferzeiger auf nächstes Byte setzen
A832    CA         DEX            Zeiger auf nächsten Spureintrag
A833    10 F7      BPL $A82C      schon alle Spuren eingetragen ?
A835    A0 EE      LDY #$EE       ja, Zeiger auf Spur 18, Seite 2 richten
A837    A9 00      LDA #$00       Zahl der freien Blocks der Spur
A839    91 6D      STA ($6D),Y    löschen (da Directoryspur)
A83B    4C 75 D0   JMP $D075      freie Blocks der Diskette berechnen
---------------------------------------------------------------------------
[EF5F]
Sektor in BAM freigeben
A83E    AD 0F 18   LDA $180F      ja, Steuerregister holen
A841    29 20      AND #$20       und Betriebsmodus testen
A843    D0 06      BNE $A84B      ist Floppy im 1541 Modus ?
A845[1] 20 CF EF   JSR $EFCF      ja, Zeiger auf Bit eines Sektors setzen
A848    4C 62 EF   JMP $EF62      Sektor freigeben
A84B[1] A5 80      LDA $80        Nummer der aktuellen Spur holen und
```

```
A84D  C9 24     CMP #$24      mit maximalem Wert der Seite vergleichen
A84F  90 F4     BCC $A845     ist Spurnummer kleiner ?
A851  20 A1 A4  JSR $A4A1     nein, Zeiger auf BAM-Bit des Sektors
A854  20 E7 A4  JSR $A4E7     BAM-Bit des Sektors holen
A857  D0 19     BNE $A872     ist Sektor frei ?
A859  1D E9 EF  ORA $EFE9,X   ja, BAM-Bit setzen
A85C  99 46 01  STA $0146,Y   und in Puffer schreiben (freigeben)
A85F  20 88 EF  JSR $EF88     Flag für 'BAM ungültig' setzen
A862  20 08 A5  JSR $A508     Zahl der freien Blocks erhöhen
A865  A5 80     LDA $80       Nummer der aktuellen Spur auf
A867  C9 35     CMP #$35      Spur 18, Seite 2 (Directory) prüfen
A869  F0 08     BEQ $A873     identisch ?
A86B  A5 7F     LDA $7F       nein, Nummer des aktuellen Laufwerks
A86D  0A        ASL A         holen und verdoppeln
A86E  AA        TAX           (da Tabelle aus 2-Byte-Werten besteht)
A86F  4C 7F EF  JMP $EF7F     Zahl der freien Blocks der Disk erhöhen
A872  38        SEC           Fehlerflag 'Sektor schon freigegeben'
A873  60        RTS           zurück zur aufrufenden Routine
----------------------------------------------------------------------
[EF93]
Sektor in BAM belegen
A874  AD 0F 18  LDA $180F     Steuerregister holen
A877  29 20     AND #$20      und Betriebsmodus testen
A879  D0 06     BNE $A881     ist Floppy im 1541 Modus ?
A87B  20 CF EF  JSR $EFCF     Zeiger auf BAM-Bit des Sektors setzen
A87E  4C 96 EF  JMP $EF96     Sektor in BAM freigeben
A881  A5 80     LDA $80       Nummer der gewünschten Spur holen
A883  C9 24     CMP #$24      und auf max. Wert +1 für 1 Seite prüfen
A885  90 F4     BCC $A87B     ist Spur auf Seite 2 ?
A887  20 A1 A4  JSR $A4A1     ja, BAM-Zeiger auf Spureintrag setzen
A88A  20 E7 A4  JSR $A4E7     BAM-Bit des Sektors holen
A88D  F0 19     BEQ $A8A8     ist Sektor freigegeben ?
A88F  5D E9 EF  EOR $EFE9,X   ja, Sektor belegen (Bit = 0)
A892  99 46 01  STA $0146,Y   und BAM Muster wieder speichern
A895  20 88 EF  JSR $EF88     Flag für 'BAM ungültig' setzen
A898  20 1E A5  JSR $A51E     freien Blocks der Spur erniedrigen
A89B  A5 80     LDA $80       Nummer der angewählten Spur holen und
A89D  C9 35     CMP #$35      auf Spur 18, Seite 2 prüfen
A89F  F0 07     BEQ $A8A8     identisch ?
A8A1  A5 7F     LDA $7F       nein, Nummer des aktuellen Laufwerks
A8A3  0A        ASL A         holen und verdoppeln
```

```
A8A4  AA        TAX           (da Blocktabelle aus 2-Bytes besteht)
A8A5  4C B2 EF  JMP $EFB2     Zahl der freien Blocks erniedrigen
A8A8[2] 60      RTS           zurück zur aufrufenden Routine
-----------------------------------------------------------------
[F1FA]
nächsten freien Sektor der Spur suchen
A8A9  AD 0F 18  LDA $180F     Steuerregister holen
A8AC  29 20     AND #$20      und Betriebsmodus testen
A8AE  D0 06     BNE $A8B6     ist Floppy im 1541 Modus ?
A8B0[1] 20 11 F0 JSR $F011    ja, BAM-Zeiger setzen
A8B3  4C FD F1  JMP $F1FD     nächsten freien Sektor suchen
A8B6[1] A5 80   LDA $80       Nummer der aktuellen Spur
A8B8  C9 24     CMP #$24      auf max. Spur +1 der 1. Seite prüfen
A8BA  90 F4     BCC $A8B0     ist Spur auf Seite 2 ?
A8BC  20 A1 A4  JSR $A4A1     Zeiger auf BAM-Eintrag der Spur setzen
A8BF  20 45 A5  JSR $A545     Zahl der freien Blocks der Spur prüfen
A8C2  B9 2C 94  LDA $942C,Y   Zahl der Sektoren der Spur holen
A8C5  8D 4E 02  STA $024E     und merken
A8C8[1] A5 81   LDA $81       Nummer des aktuellen Sektors
A8CA  CD 4E 02  CMP $024E     mit max. Sektornummer vergleichen
A8CD  B0 09     BCS $A8D8     ist Sektornummer im erlaubten Bereich ?
A8CF  20 E7 A4  JSR $A4E7     ja, BAM-Bit des Sektors holen
A8D2  D0 06     BNE $A8DA     ist Sektor frei ?
A8D4  E6 81     INC $81       nein, nächsten Sektor anwählen
A8D6  D0 F0     BNE $A8C8     immer Sprung nach $A8C8
A8D8[1] A9 00   LDA #$00      Flag für 'kein Sektor frei'
A8DA[1] 60      RTS           zurück zur aufrufenden Routine
-----------------------------------------------------------------
[F12D]
nächsten freien Sektor suchen
A8DB  AD 0F 18  LDA $180F     Steuerregister holen
A8DE  29 20     AND #$20      und Betriebsmodus testen
A8E0  D0 06     BNE $A8E8     ist Floppy im 1541 Modus ?
A8E2[1] A5 6F   LDA $6F       ja, Zahl der freien Blocks der Spur
A8E4  48        PHA           retten
A8E5  4C 30 F1  JMP $F130     nächsten freien Sektor suchen
A8E8[1] A5 80   LDA $80       Nummer der aktuellen Spur
A8EA  C9 24     CMP #$24      auf max. Spur +1 der 1. Seite prüfen
A8EC  90 F4     BCC $A8E2     ist Spur auf Seite 2 ?
A8EE  C9 35     CMP #$35      ja, auf Spur 18, Seite 2 testen
A8F0  F0 0E     BEQ $A900     identisch ?
```

```
A8F2  A5 6F      LDA $6F         Zeropagestelle
A8F4  48         PHA             retten
A8F5  20 A1 A4   JSR $A4A1       Zeiger auf BAM-Muster setzen
A8F8  A8         TAY             Zahl der freien Blocks merken
A8F9  68         PLA             Zeropagestelle wieder
A8FA  85 6F      STA $6F         herstellen
A8FC  98         TYA             Zahl der freien Blocks der Spur holen
A8FD  4C 38 F1   JMP $F138       freien Sektor holen
A900  A9 00      LDA #$00        Zahl der freien Blocks der Spur setzen
A902  4C 38 F1   JMP $F138       nächsten freien Sektor suchen
-----------------------------------------------------------------

[F1C4]
BAM Zeiger auf Bit eines Sektors setzen
A905  AD 0F 18   LDA $180F       Steuerregister holen
A908  29 20      AND #$20        und Betriebsmodus testen
A90A  D0 06      BNE $A912       ist Floppy im 1541 Modus ?
A90C  20 11 F0   JSR $F011       ja, BAM-Zeiger setzen und
A90F  4C C7 F1   JMP $F1C7       optimalen freien Sektor holen
A912  A5 80      LDA $80         Nummer der aktuellen Spur
A914  C9 24      CMP #$24        auf max. Spur +1 der 1. Seite prüfen
A916  90 F4      BCC $A90C       ist Spur auf Seite 2 ?
A918  20 A1 A4   JSR $A4A1       BAM-Zeiger setzen
A91B  4C C9 F1   JMP $F1C9       optimalen freien Sektor holen
-----------------------------------------------------------------

[F1D1]
freien Sektor suchen
A91E  AD 0F 18   LDA $180F       Steuerregister holen
A921  29 20      AND #$20        und Betriebsmodus testen
A923  D0 06      BNE $A92B       ist Floppy im 1541 Modus ?
A925  20 11 F0   JSR $F011       ja, BAM-Zeiger setzen
A928  4C E2 F1   JMP $F1E2       freien Sektor suchen
A92B  A5 80      LDA $80         Nummer der aktuellen Spur
A92D  C9 24      CMP #$24        auf max. Spur +1 der 1. Seite prüfen
A92F  90 F4      BCC $A925       ist Spur auf Seite 2 ?
A931  20 A1 A4   JSR $A4A1       Zeiger auf BAM-Bit setzen
A934  4C E4 F1   JMP $F1E4       freien Sektor suchen
-----------------------------------------------------------------

[EF28]
Zahl der freien Blocks in BAM prüfen
A937  AD 0F 18   LDA $180F       Steuerregister holen
A93A  29 20      AND #$20        und Betriebsmodus testen
```

```
A93C  D0 03     BNE $A941       ist Floppy im 1541 Modus ?
A93E  4C 20 F2  JMP $F220       ja, Blockangaben prüfen
A941  AD AC 02  LDA $02AC       Nummer der größten Spur holen und
A944  C9 25     CMP #$25        mit 37 vergleichen
A946  90 F6     BCC $A93E       nur eine Diskseite verwendet ?
A948  A5 80     LDA $80         nein, Nummer der aktuellen Spur holen
A94A  C9 24     CMP #$24        und mit 36 vergleichen
A94C  90 F0     BCC $A93E       ist Spur auf Seite 2 ?
A94E  4C 45 A5  JMP $A545       ja, Blockangaben überprüfen
-----------------------------------------------------------------------
```

[D097]

Zahl der freien Blocks der Diskette berechnen

```
A951  9D FA 02  STA $02FA,X     Low-Byte der freien Blocks speichern
A954  AD 0F 18  LDA $180F       Steuerregister holen
A957  29 20     AND #$20        und Betriebsmodus testen
A959  F0 23     BEQ $A97E       ist Floppy im 1571 Modus ?
A95B  AD AC 02  LDA $02AC       ja, maximale Spurnummer feststellen
A95E  C9 25     CMP #$25        und mit max Spur +2 verleichen (37)
A960  90 1C     BCC $A97E       ist Spur auf Seite 2 ?
A962  20 8B A4  JSR $A48B       aktuellen BAM-Zeiger retten
A965  20 34 A5  JSR $A534       BAM-Zeiger auf Spureintrag setzen
A968  A0 22     LDY #$22        Zahl der Spuren auf einer Seite -1
A96A  AD FA 02  LDA $02FA       Low-Byte der freien Blocks holen
A96D  18        CLC             und Byte das freie Blocks der Spuur
A96E  71 6D     ADC ($6D),Y     angibt einrechnen
A970  8D FA 02  STA $02FA       neue Blockzahl merken
A973  90 03     BCC $A978       ist ein Übertrag entstanden ?
A975  EE FC 02  INC $02FC       ja, High-Byte der 'Blocks free' setzen
A978  88        DEY             nächste Spur anwählen
A979  10 F2     BPL $A96D       alle Spuren berücksichtigt ?
A97B  4C 96 A4  JMP $A496       ja, BAM-Zeiger auf alten Wert setzen
A97E  60        RTS             zurück zur aufrufenden Routine
-----------------------------------------------------------------------
```

[DCD6]

Patch (=Korrektur) für Zahl der freien Blocks und Zeiger löschen

```
A97F  95 B5     STA $B5,X       Zahl der belegten Blöcke einer Datei
A981  95 BB     STA $BB,X       (Low- und Highbyte) löschen
A983  A9 00     LDA #$00        Zahl der noch zu übertragenden
A985  9D 44 02  STA $0244,X     Datenbytes löschen
A988  60        RTS             zurück zur aufrufenden Routine
-----------------------------------------------------------------------
```

```
[84E4]
Diskette im 1571 Format formatieren
A989  20 0D A4   JSR $A40D       Diskette formatieren
A98C  A0 00      LDY #$00        Flag für 'Fehler beachten'
A98E  8C 98 02   STY $0298       setzen
A991  60         RTS             zurück zur aufrufenden Routine
-----------------------------------------------------------------------
A992  FF ...                     unbenutzter
A99C  ... FF                     ROM-Bereich
-----------------------------------------------------------------------
[C1B3]  Patch für 1541 Routine (Fehler bei FF,X behoben)
A99D  A9 00      LDA #$00        Laufwerksstatus
A99F  9D FF 00   STA $00FF,X     für 'Laufwerk bereit' setzen
A9A2  4C B7 C1   JMP $C1B7       zurück zur 1541 Routine
-----------------------------------------------------------------------
[C661]  Patch für 1541 Routine (Fehler bei FF,X behoben)
A9A5  98         TYA             Rückmeldung holen
A9A6  9D FF 00   STA $00FF,X     und in Laufwerksstatus übertragen
A9A9  4C 64 C6   JMP $C664       zurück zur 1541 Routine
-----------------------------------------------------------------------
[EA6B]
seriellen Bus nach ATN-Kommando bearbeiten
A9AC  AD 0F 18   LDA $180F       Steuerregister holen
A9AF  29 20      AND #$20        und auf Betriebsmodus testen
A9B1  F0 03      BEQ $A9B6       ist Floppy im 1571 Modus ?
A9B3  4C 5A 81   JMP $815A       ja, 1571 Bus bediehnen
A9B6  4C D7 E8   JMP $E8D7       1541 Bus bearbeiten
-----------------------------------------------------------------------
[E60A]
Fehlermeldung ausgeben und Klartextmeldung bereitsstellen
A9B9  48         PHA             Fehlernummer retten
A9BA  86 F9      STX $F9         Nummer des Puffers merken
A9BC  AD 0F 18   LDA $180F       Steuerregister holen
A9BF  29 20      AND #$20        und auf Betriebsmodus testen
A9C1  F0 0F      BEQ $A9D2       ist Floppy im 1571 Modus ?
A9C3  24 37      BIT $37         ja, Busstatusbyte auf 1571 Modus prüfen
A9C5  10 0B      BPL $A9D2       Flag gesetzt ?
A9C7  A5 37      LDA $37         ja, Flag für
A9C9  29 7F      AND #$7F        1571 Modus im Busstatusbyte
A9CB  85 37      STA $37         löschen
A9CD  68         PLA             Fehlernummer wieder holen
```

```
A9CE  AA          TAX            und für Ausgabe vorbereiten
A9CF  4C 99 91    JMP $9199      Nummer auf 1571 Bus und Meldung erzeugen
A9D2  4C 0D E6    JMP $E60D      Klartextmeldung bereitstellen
-----------------------------------------------------------------------
[C1CE]
Fehlermeldung im Fehlerpuffer erzeugen
A9D5  48          PHA            Nummer des Fehlers retten
A9D6  AD 0F 18    LDA $180F      Steuerregister holen
A9D9  29 20       AND #$20       und auf Betriebsmodus testen
A9DB  F0 17       BEQ $A9F4      ist Floppy im 1571 Modus ?
A9DD  24 37       BIT $37        ja, Busstatusbyte auf 1571 Modus prüfen
A9DF  10 13       BPL $A9F4      Flag gesetzt ?
A9E1  A5 37       LDA $37        ja, Flag für
A9E3  29 7F       AND #$7F       1571 Modus im Busstatusbyte
A9E5  85 37       STA $37        löschen
A9E7  78          SEI            Bus- und Kontrolleraufruf abbschalten
A9E8  A2 02       LDX #$02       Fehlernummer für 'File Not Found'
A9EA  20 28 92    JSR $9228      auf 1571 Bus ausgeben
A9ED  A9 00       LDA #$00       Sekundäradresse für Load
A9EF  85 83       STA $83        setzen
A9F1  20 C0 DA    JSR $DAC0      Datei schließen
A9F4  68          PLA            Fehlernummer wieder holen
A9F5  4C 45 E6    JMP $E645      Fehlermeldung erzeugen
-----------------------------------------------------------------------
[F263]  Patch für 1541 Routine (neu: Status löschen)
A9F8  A9 00       LDA #$00       Laufwerkstatus für Laufwerk 0
A9FA  85 20       STA $20        löschen
A9FC  AD 0C 1C    LDA $1C0C      Peripheres Steuerregister holen
A9FF  4C 66 F2    JMP $F266      zurück zur 1541 Routine
-----------------------------------------------------------------------
[C2BA/C2C2]
neuen User0-Befehl beachten
AA02  AD 00 02    LDA $0200      1. Zeichen aus Befehlsstring holen
AA05  C9 55       CMP #$55       und mit 'U' (User-Befehl) vergleichen
AA07  D0 07       BNE $AA10      identisch ?
AA09  AD 01 02    LDA $0201      ja, 2. Zeichen aus Befehlsstring holen
AA0C  C9 30       CMP #$30       und auf '0' prüfen
AA0E  F0 04       BEQ $AA14      ist Befehl 'U0' ?
AA10  B9 00 02    LDA $0200,Y    nein, Zeichen aus Befehlsstring holen
AA13  2C          .byte $2C      nächste 2 Bytes überspringen (Bitbefehl)
AA14  A9 00       LDA #$00       Leerparameter bei User0 zurückgeben
```

```
AA16   60          RTS              zurück zur aufrufenden Routine
--------------------------------------------------------------------
[C66B]  Patch für 1541 Routine (Fehler bei FF,X behoben)
AA17   A6 7F       LDX $7F          Nummer des aktuellen Laufwerks
AA19   BD FF 00    LDA $00FF,X      dazugehörigen Laufwerksstatus holen
AA1C   60          RTS              zurück zur aufrufenden Routine
--------------------------------------------------------------------
[D071]  Patch für 1541 Routine (Fehler bei FF,X behoben)
AA1D   95 1C       STA $1C,X        Flag für Diskette initialisieren setzen
AA1F   9D FF 00    STA $00FF,X      Laufwerksstatus festlegen
AA22   4C 75 D0    JMP $D075        zurück zur 1541 Routine
--------------------------------------------------------------------
[F017]  Patch für 1541 Routine (Fehler bei FF,X behoben)
AA25   A6 7F       LDX $7F          Nummer des aktuellen Laufwerks
AA27   BD FF 00    LDA $00FF,X      dazugehörigen Laufwerksstatus holen
AA2A[1] 4C 1B F0   JMP $F01B        zurück zur 1541 Routine
--------------------------------------------------------------------
[CB81]
User-Befehle (UA bis UK) ausführen
AA2D   A5 75       LDA $75          Low-Byte der User-Routine holen
AA2F   C9 67       CMP #$67         und auf IRQ prüfen
AA31   D0 09       BNE $AA3C        identisch ?
AA33   A5 76       LDA $76          ja, High-Byte holen und
AA35   C9 FE       CMP #$FE         mit Adresse für IRQ vergleichen
AA37   D0 03       BNE $AA3C        identisch ?
AA39   00          BRK              ja, Jobschleife aufrufen
AA3A   EA          NOP              Rücksprungadresse ausgleichen
AA3B   60          RTS              zurück zur aufrufenden Routine
AA3C[2] 6C 75 00   JMP ($0075)      User-Befehl ausführen
--------------------------------------------------------------------
AA3F   FF ...                       unbenutzter
BEFF   ... FF                       ROM-Bereich
--------------------------------------------------------------------
[Tabelle wird im DOS nicht verwendet]
Tabelle der wichtigsten DOS-Routinen
BF00   4C 88 9D    JMP $9D88        1541 IRQ Routine
BF03   4C DE 9D    JMP $9DDE        1571 IRQ Routine
BF06   4C B0 F2    JMP $F2B0        1541 Jobschleife
BF09   4C BA 92    JMP $92BA        1571 Jobschleife
BF0C   4C 93 F3    JMP $F393        Pufferzeiger für Jobschleife setzen
BF0F   4C D1 93    JMP $93D1        Pufferzeiger für Jobschleife setzen
```

```
BF12  4C 69 F9   JMP $F969       Job abschließen; Rückmeldung übergeben
BF15  4C B5 99   JMP $99B5       Job abschließen; Rückmeldung übergeben
BF18  4C 00 FE   JMP $FE00       Kopf auf Lesen umschalten
BF1B  4C 34 F9   JMP $F934       Blockheader in GCR-Werte umrechnen
BF1E  4C 56 F5   JMP $F556       auf Sync-Markierung warten  (1541)
BF21  4C 54 97   JMP $9754       auf Sync-Markierung warten  (1571)
BF24  4C E0 F8   JMP $F8E0       Zusatzpuffer von GCR nach Binär wandeln
BF27  4C 65 99   JMP $9965       Zusatzpuffer von GCR nach Binär wandeln
BF2A  4C E9 F5   JMP $F5E9       Prüfsumme des Sektors berechnen
BF2D  4C E6 F7   JMP $F7E6       5 GCR-Bytes in 4 Binärbyte umwandeln
BF30  4C D9 98   JMP $98D9       5 GCR-Bytes in 4 Binärbyte umwandeln
BF33  4C 83 A4   JMP $A483       ca. 80 Taktzyklen warten
BF36  4C F3 FE   JMP $FEF3       Verzögerung für seriellen 1541-Bus
BF39  4C 7E A4   JMP $A47E       ca. 45 Taktzyklen warten
BF3C  4C 05 F0   JMP $F005       Puffer für BAM löschen
BF3F  4C D1 F0   JMP $F0D1       Spurnummer für BAM holen
BF42  4C 46 C1   JMP $C146       Befehlsstring ausführen
BF45  4C 68 C2   JMP $C268       Befehlsstring auf Parameter untersuchen
BF48  4C B3 C2   JMP $C2B3       Zeiger für Befehlstringanalyse setzen
BF4B  4C DC C2   JMP $C2DC       alle Dateizeiger löschen
BF4E  4C E6 86   JMP $86E6       Routine mit Nr. in Akku ausführen
BF51  4C 64 87   JMP $8764       Laufwerksmotor einschalten
BF54  4C 70 87   JMP $8770       Laufwerksmotor ausschalten
BF57  4C 8E 80   JMP $808E       [Fehler siehe 7.1.5]
BF5A  4C 1E CF   JMP $CF1E       Puffer suchen und setzen
BF5D  4C B4 D7   JMP $D7B4       Open Befehl vom Bus ausführen
BF60  4C C0 DA   JMP $DAC0       Kanal und Datei schließen
BF63  4C 0A E6   JMP $E60A       Fehlermeldung der Jobschleife  ausgeben
BF66  4C 80 90   JMP $9080       Datei einlesen (PRG/SEQ/USR)
BF69  4C 4E 92   JMP $924E       Prüfsumme des ROMs testen
BF6C  4C 59 F2   JMP $F259       1541-Diskkontroller initialisieren
BF6F  4C 9C F9   JMP $F99C       1541-Jobschleife abschließen
BF72  4C CA 99   JMP $99CA       Steppersteuerrung
BF75  4C 95 FE   JMP $FE95       [Fehler siehe 7.1.5]
------------------------------------------------------------------
BF78  FF ...                     unbenutzter
C0FF  ... FF                     ROM-Bereich
------------------------------------------------------------------
```

```
[C38C/C439/C46A/C49A/CE0B/CFA2/D39E/D7E1]  vgl. 877C/C118
LED am aktuellen Laufwerk einschalten
(Routine aus alten Doppellaufwerken entnommen)
C100  78        SEI          Diskcontroller abschalten
C101  A9 F7     LDA #$F7     Maske für LED-Bit (Bit3)
C103  2D 00 1C  AND $1C00    erstellen, wobei LED-Bit gelöscht wird
C106  48        PHA          Maske merken
C107  A5 7F     LDA $7F      Laufwerknummer (immer 0)
C109  F0 05     BEQ $C110    daher immer Sprung nach $C110
C10B  68        PLA          unbenutzt, da die Station keine
C10C  09 00     ORA #$00     zwei Laufwerke hat
C10E  D0 03     BNE $C113    bei Laufwerk 1, immer Sprung nach $C113
C110  68        PLA          Maske wieder holen
C111  09 08     ORA #$08     LED-Bit setzen (Bit3=1)
C113  8D 00 1C  STA $1C00    LED an
C116  58        CLI          Diskcontroller wieder einschalten
C117  60        RTS          Zurück zur aufrufenden Routine
-----------------------------------------------------------------
[Routine wird im DOS nicht verwendet]  vgl. 877C/C100
LED am 1571-Laufwerk einschalten
C118  78        SEI          Diskcontroller abschalten
C119  A9 08     LDA #$08     LED-Bit (Bit3) setzen (Bit3=1)
C11B  0D 00 1C  ORA $1C00    andere Bits des Registers übernehmen
C11E  8D 00 1C  STA $1C00    LED an
C121  58        CLI          Diskcontroller wieder einschalten
C122  60        RTS          zurück zur aufrufenden Routine
-----------------------------------------------------------------
[C1AA/D425/E6BC]
Fehlerflags löschen
C123  A9 00     LDA #$00     Löschen der Fehlerflags
C125  8D 6C 02  STA $026C    Fehlernummer loeschen
C128  8D 6D 02  STA $026D    Flag für LED-Blinken löschen
C12B  60        RTS          zurück zur aufrufenden Routine
-----------------------------------------------------------------
[E650]
LED-Blinken bei Fehler einschalten
C12C  78        SEI          Diskcontroller ausschalten
C12D  8A        TXA          X-Register
C12E  48        PHA          retten
C12F  A9 50     LDA #$50     LED-Blinkzähler auf 80
C131  8D 6C 02  STA $026C    setzen
```

```
C134  A2 00     LDX #$00       Laufwerk 0 anwählen
C136  BD CA FE  LDA $FECA,X    LED-Maske für angewähltes Laufwerk
C139  8D 6D 02  STA $026D      merken
C13C  0D 00 1C  ORA $1C00      LED am Laufwerk
C13F  8D 00 1C  STA $1C00      einschalten
C142  68        PLA            X-Register
C143  AA        TAX            zurücksetzten
C144  58        CLI            Diskcontroller einschalten
C145  60        RTS            zurück zur aufrufenden Routine
-----------------------------------------------------------------------
[A61C/BF42]
Befehlsstrings vom Rechner ausführen
C146  A9 00     LDA #$00       Flag für 'BAM auf Diskette schreiben'
C148  8D F9 02  STA $02F9      setzen
C14B  AD 8E 02  LDA $028E      letztes benutztes Laufwerk
C14E  85 7F     STA $7F        als aktuelles Laufwerk übernehmen
C150  20 BC E6  JSR $E6BC      'OK'-Meldung erzeugen
C153  A5 84     LDA $84        letzte IEC Sekundäradresse holen
C155  10 09     BPL $C160      war das ein Close Kommando ?
C157  29 0F     AND #$0F       Nummer des angewählten Kanals holen
C159  C9 0F     CMP #$0F       und auf Kommandokanal prüfen
C15B  F0 03     BEQ $C160      ist Kommandokanal angeesprochen ?
C15D  4C B4 D7  JMP $D7B4      nein,
C160  20 B3 C2  JSR $C2B3      Parameter zur Befehlsbearbeitung setzen
C163  B1 A3     LDA ($A3),Y    erstes Zeichen aus Eingabepuffer
C165  8D 75 02  STA $0275      holen und merken
C168  A2 0B     LDX #$0B       Zahl der Floppy-Befehle
C16A  BD 89 FE  LDA $FE89,X    Zeichen aus 1541-Kommandotabelle holen
C16D  CD 75 02  CMP $0275      und mit Befehlszeichen vergleichen
C170  F0 08     BEQ $C17A      ist das der gewünschte Befehl ?
C172  CA        DEX            nein, nächstes Kommandozeichen anwählen
C173  10 F5     BPL $C16A      mit allen Kommandos verglichen ?
C175  A9 31     LDA #$31       ja, Fehlermeldung
C177  4C C8 C1  JMP $C1C8      '31 Syntax Error' ausgeben
C17A  8E 2A 02  STX $022A      Nummer des Befehls merken
C17D  E0 09     CPX #$09       mit Nummer für 'Rename' vergleichen
C17F  90 03     BCC $C184      ist Befehl 'R', 'S' oder 'N' ?
C181  20 EE C1  JSR $C1EE      ja, Syntax prüfen
C184  AE 2A 02  LDX $022A      Nummer des Kommandos zurückholen
C187  BD 95 FE  LDA $FE95,X    Low-Byte des Befehls aus Tabelle holen
C18A  85 6F     STA $6F        und in Zeiger setzen
```

```
C18C  BD A1 FE   LDA $FEA1,X      High-Byte der Startadresse
C18F  85 70      STA $70          übertragen
C191  6C 6F 00   JMP ($006F)      Befehl ausführen
-----------------------------------------------------------------
[9196/C99E/C9A4/CAC9/CB6F/CC18/CCFB/CD16/CD5C/CD70/CD94/CDA0/CDB7/CDCF]
[D00B/D7F0/D99D/DAE6/DAFC/E272/EDB0/EEB4]
Beenden eines Rechnerbefehls, erzeugen der Fehlermeldung
C194  A9 00      LDA #$00         Flag für 'BAM auf Diskette schreiben'
C196  8D F9 02   STA $02F9        löschen
-----------------------------------------------------------------
[DA06]  Befehlsende; aber BAM nicht auf Diskette schreiben
C199  AD 6C 02   LDA $026C        Fehlerflag holen
C19C  D0 2A      BNE $C1C8        ist ein Fehler aufgetreten ?
C19E  A0 00      LDY #$00         nein, OK-Meldung bereitstellen
C1A0  98         TYA              Fehlernummer löschen
-----------------------------------------------------------------
[C87A]  Befehlsende; Fehler ignorieren
C1A1  84 80      STY $80          Spur und
C1A3  84 81      STY $81          Sektor auf Null setzen
C1A5  84 A3      STY $A3          Zeiger auf Eingabepuffer zurücksetzen
C1A7  20 C7 E6   JSR $E6C7        'OK'-Meldung erzeugen
C1AA  20 23 C1   JSR $C123        Fehlerflags löschen
-----------------------------------------------------------------
[DAE9/DAFF]  Befehlsende; keine Rückmeldung bereitstellen
C1AD  A5 7F      LDA $7F          aktuelles Laufwerk (immer 0)
C1AF  8D 8E 02   STA $028E        als letzte Drivenummer merken
C1B2  AA         TAX              für aktuelles Laufwerk
C1B3  4C 9D A9   JMP $A99D        Flag für 'Laufwerk aktiv' löschen
C1B6  EA         NOP              [durch Modifikation des 1541 ROM]
C1B7  20 BD C1   JSR $C1BD        Eingabepuffer löschen ($0200-$0228)
C1BA  4C DA D4   JMP $D4DA        interne Lese/Schreibkanäle schließen
-----------------------------------------------------------------
[C1B7/E648]
Eingabepuffer für Befehle vom Rechner löschen
C1BD  A0 28      LDY #$28         41 Zeichenpositionen
C1BF  A9 00      LDA #$00         mit Null überschreiben
C1C1  99 00 02   STA $0200,Y      (Bereich $0200-$0228)
C1C4  88         DEY              nächstes Zeichen
C1C5  10 FA      BPL $C1C1        Schon alle Zeichen gelöscht ?
C1C7  60         RTS              ja, zurück zur aufrufenden Routine
-----------------------------------------------------------------
```

```
[850E/85AC/9023/91C7/C177/C19C/C1F5/C265/C2D9/C41D/C8C3/C925/C984/CAE3]
[CAF1/CB4D/CBA2/CC28/CC2D/CD33/CDE2/CF78/CFFA/D214/D38E/D839/D8ED/D8F2]
[D947/D959/D967/D9C0/E0D3/E16B/E216/E225/E299/E365/E44B/E7C2/EE16/F0EF]
[F15C]
Fehlermeldung ausgeben, Nummer im Akku erforderlich
C1C8  A0 00     LDY #$00      Zeiger löschen
C1CA  84 80     STY $80       Spurnummer und
C1CC  84 81     STY $81       Sektornummer löschen
C1CE  4C D5 A9  JMP $A9D5     Meldung im Puffer erstellen
--------------------------------------------------------------------
[D005/D7FF/ED84]
Sucht ':' und Laufwerksnummer im Befehlsstring
(Y-Register muß auf aktuelle Position im Puffer zeigen)
C1D1  A2 00     LDX #$00      Zeiger auf Position der Laufwerks-
C1D3  8E 7A 02  STX $027A     nummer im Eingabepuffer löschen
C1D6  A9 3A     LDA #$3A      Doppelpunkt als zu suchendes Zeichen
C1D8  20 68 C2  JSR $C268     sucht Zeichen im Eingabepuffer
C1DB  F0 05     BEQ $C1E2     wurde Doppelpunkt gefunden ?
C1DD  88        DEY           ja, Y-Register zeigt auf Position +1 des
C1DE  88        DEY           Zeichens;
C1DF  8C 7A 02  STY $027A     Position der Laufwerksnummer (vor ':')
C1E2[1] 4C 68 C3  JMP $C368   Laufwerk setzen und LED anschalten
--------------------------------------------------------------------
[C1EE/C904/D82B/DA86]
Sucht Doppelpunkt im Befehlsstring
C1E5  A0 00     LDY #$00      Startposition der Suche auf Anfang
C1E7  A2 00     LDX #$00      Anzahl der gefundenen Filenamen
C1E9  A9 3A     LDA #$3A      Doppelpunkt als zu suchendes Zeichen
C1EB  4C 68 C2  JMP $C268     Eingabepuffer durchsuchen
--------------------------------------------------------------------
[C181]
Prüft bei Befehlen mit 2 Dateinennungen auf Syntax
C1EE  20 E5 C1  JSR $C1E5     Sucht Doppelpunkt im Eingabepuffer
C1F1  D0 05     BNE $C1F8     wurde Doppelpunkt gefunden ?
C1F3[1] A9 34     LDA #$34    nein, Fehlermeldung
C1F5  4C C8 C1  JMP $C1C8     '34 Syntax Error' ausgeben
C1F8[2] 88        DEY         Y-Register zeigt auf Position +1
C1F9  88        DEY           Zeiger auf Zeichen vor ':' setzen
C1FA  8C 7A 02  STY $027A     Position der Laufwerksnummer merken
C1FD  8A        TXA           Zahl der bereits gefundenen Dateinamen
C1FE  D0 F3     BNE $C1F3     wurden mehrere Namen gefunden ?
```

```
C200[1] A9 3D      LDA #$3D      nein, '=' - Zeichen
C202    20 68 C2   JSR $C268     Zeile nach '=' durchsuchen
C205    8A         TXA           Zahl der dabei gefundenen Dateinamen
C206    F0 02      BEQ $C20A     ist nur eine Datei gefunden worden ?
C208    A9 40      LDA #$40      nein, Bit 6 als Flag für weitere Dateien
C20A[1] 09 21      ORA #$21      Bit 0 und 5 für '1. Dateinamen gefunden'
C20C    8D 8B 02   STA $028B     Bitflags merken
C20F    E8         INX           zeigt auf Ende des 1.Dateinamens
C210    8E 77 02   STX $0277     Anzahl der Dateien, die für die erste
C213    8E 78 02   STX $0278     Dateibeschreibung gefunden wurden
C216    AD 8A 02   LDA $028A     Flag für Joker ('*')
C219    F0 0D      BEQ $C228     Joker im Dateinamen vorhanden ?
C21B    A9 80      LDA #$80      ja, Flag im Syntaxbyte
C21D    0D 8B 02   ORA $028B     setzen und
C220    8D 8B 02   STA $028B     Flag wieder merken
C223    A9 00      LDA #$00      Jokerflag der Suchroutine
C225    8D 8A 02   STA $028A     löschen
C228[1] 98         TYA           Position des '=' - Zeichens im Befehl
C229    F0 29      BEQ $C254     Ende der Befehlszeile gefunden ?
C22B    9D 7A 02   STA $027A,X   nein, Position des Dateinamens merken
C22E    AD 77 02   LDA $0277     Zahl der Dateien für erste Dateinennung
C231    8D 79 02   STA $0279     als Zahl für zweite Nennung setzen
C234    A9 8D      LDA #$8D      Endekennzeichen des Befehlsstrings und
C236    20 68 C2   JSR $C268     bis zum Ende weiter untersuchen
C239    E8         INX           Zahl der gefundenen Kommas und somit
C23A    8E 78 02   STX $0278     Zahl der weiteren Dateinamen merken
C23D    CA         DEX           ursprünglichen Wert herstellen
C23E    AD 8A 02   LDA $028A     Flag für Joker ('*')
C241    F0 02      BEQ $C245     Joker vorhanden ?
C243    A9 08      LDA #$08      ja, Bit 3 als Flag setzen
C245[1] EC 77 02   CPX $0277     Länge mit altem Wert vergleichen
C248    F0 02      BEQ $C24C     weitere Dateinamen gefunden ?
C24A    09 04      ORA #$04      ja, Flag für Dateinamen nach '=' Zeichen
C24C[1] 09 03      ORA #$03      Flag für '=' Zeichen vorhanden
C24E    4D 8B 02   EOR $028B     bisherige Flags einblenden und
C251    8D 8B 02   STA $028B     als neuen Syntaxstatus merken
C254[1] AD 8B 02   LDA $028B     Syntaxflag für Befehl
C257    AE 2A 02   LDX $022A     Nummer des Befehls
C25A    3D A5 FE   AND $FEA5,X   vorhandene mit erlaubten Syntaxelementen
C25D    D0 01      BNE $C260     vergleichen; alle erlaubt ?
C25F    60         RTS           ja, zurück zur aufrufenden Routine
```

```
C260[1] 8D 6C 02  STA $026C      Art der fehlerhaften Syntax merken
C263    A9 30     LDA #$30       Fehlermeldung
C265    4C C8 C1  JMP $C1C8      '30 Syntax Error' ausgeben
------------------------------------------------------------------------------
[C1D8/C1EB/C202/C236/CC21/CC75/D845]
Untersucht die Eingabezeile auf Zeichen aus Akku
(Y-Register muss die aktuelle Position im Eingabepuffer enthalten;
(X-Register enthält danach die Zahl der gefundenen Parameter)
C268    8D 75 02  STA $0275      Zeichen nach dem gesucht werden soll
C26B[2] CC 74 02  CPY $0274      Länge des Befehlsstrings prüfen
C26E    B0 2E     BCS $C29E      Ende erreicht ?
C270    B1 A3     LDA ($A3),Y    nein, Zeichen aus Eingabepuffer holen
C272    C8        INY            Zeiger auf nächstes Zeichen setzen
C273    CD 75 02  CMP $0275      Zeichen, das gesucht wird
C276    F0 28     BEQ $C2A0      identisch mit Zeichen aus Eingabezeile ?
C278    C9 2A     CMP #$2A       nein, mit Joker ('*') vergleichen
C27A    F0 04     BEQ $C280      identisch ?
C27C    C9 3F     CMP #$3F       nein, mit Joker ('?') vergleichen
C27E    D0 03     BNE $C283      identisch ?
C280[1] EE 8A 02  INC $028A      ja, Jokerflag setzen
C283[1] C9 2C     CMP #$2C       aktuelles Zeichen mit ',' vergleichen
C285    D0 E4     BNE $C26B      identisch ?
C287    98        TYA            ja, Position des Kommas +1 als Start-
C288    9D 7B 02  STA $027B,X    position des nächsten Parameters merken
C28B    AD 8A 02  LDA $028A      Jokerflag zurückholen
C28E    29 7F     AND #$7F       Flag für Joker löschen
C290    F0 07     BEQ $C299      ist Joker aufgetreten ?
C292    A9 80     LDA #$80       ja, Bit 7 als Flag setzen und
C294    95 E7     STA $E7,X      Dateiname als Name mit Joker markieren
C296    8D 8A 02  STA $028A      Bit 7 des Jokerflag setzen
C299[1] E8        INX            Zahl der gefundenen, durch Komma
C29A    E0 04     CPX #$04       getrennten Parameter erhöhen; max. Zahl
C29C    90 CD     BCC $C26B      von 5 verarbeitbaren Dateien erreicht ?
C29E[1] A0 00     LDY #$00       ja, Y-Wert =0 signalisiert Ende
C2A0[1] AD 74 02  LDA $0274      Länge der Befehlszeile als Startposition
C2A3    9D 7B 02  STA $027B,X    des letzten Parameters  merken
C2A6    AD 8A 02  LDA $028A      Jokerflag des letzten Dateinamens holen
C2A9    29 7F     AND #$7F       Bit 7 ausblenden
C2AB    F0 04     BEQ $C2B1      Joker im Parameter vorhanden ?
C2AD    A9 80     LDA #$80       ja, Dateiname in Dateitabelle
C2AF    95 E7     STA $E7,X      als Name mit Joker kennzeichen
```

```
C2B1[1] 98        TYA           Position in Eingabezeile (bei Ende =0)
C2B2    60        RTS           zurück zur aufrufenden Routine
-----------------------------------------------------------------------
[BF48/C160/D7B9/E207/C2DC:BF4B,DA86]
Setzt alle Flags und Tabellen um Befehlsstring zu untersuchen
C2B3    A4 A3     LDY $A3       Low Byte des Eingabepuffer-Zeigers
C2B5    F0 14     BEQ $C2CB     Adresse $A3/$A4 =$0200 ?
C2B7    88        DEY           nein, Pufferzeiger korregieren
C2B8    F0 10     BEQ $C2CA     ist $A3 =1 ?
C2BA    20 02 AA  JSR $AA02     nein, auf User Befehl prüfen
C2BD    C9 0D     CMP #$0D      Zeichen aus Befehl auf Return prüfen
C2BF    F0 0A     BEQ $C2CB     Zeilenende erreicht ?
C2C1    88        DEY           nein, Zeiger zuf Zeichen davor setzen
C2C2    B9 00 02  LDA $0200,Y   Zeichen aus Puffer holen und mit Return
C2C5    C9 0D     CMP #$0D      (= Zeilenende) vergleichen
C2C7    F0 02     BEQ $C2CB     identisch ?
C2C9    C8        INY           nein, Pufferzeiger auf Ausgangswert
C2CA[1] C8        INY           zurücksetzen
C2CB[3] 8C 74 02  STY $0274     Wert des Zeigers als Ende des Befehls-
C2CE    C0 2A     CPY #$2A      strings merken; max. Länge erreicht ?
C2D0    A0 FF     LDY #$FF      Wert für Befehlsnummer auf 'kein Befehl'
C2D2    90 08     BCC $C2DC     Befehlsstring kleiner als Puffer (42) ?
C2D4    8C 2A 02  STY $022A     nein, Befehlsnummer löschen
C2D7    A9 32     LDA #$32      Zeile zu lang
C2D9    4C C8 C1  JMP $C1C8     Fehler '32 Syntax Error' ausgeben
C2DC[3] A0 00     LDY #$00      Tabellen, Zeiger und Flags löschen
C2DE    98        TYA           und zurücksetzen
C2DF    85 A3     STA $A3       Zeiger auf Eingabepuffer auf $0200
C2E1    8D 58 02  STA $0258     aktuelle Recordlänge
C2E4    8D 4A 02  STA $024A     aktueller Dateityp
C2E7    8D 96 02  STA $0296     Dateityp aus Befehlsstring
C2EA    85 D3     STA $D3       Zeiger auf ersten Dateinamen
C2EC    8D 79 02  STA $0279     Zeiger auf Dateinamen
C2EF    8D 77 02  STA $0277     Zahl der Dateien für 1. Dateinennung
C2F2    8D 78 02  STA $0278     Zahl der Dateien für 2. Dateinennung
C2F5    8D 8A 02  STA $028A     Flag für Joker im Dateinamen aufgetreten
C2F8    8D 6C 02  STA $026C     Flag für Synatxfehler
C2FB    A2 05     LDX #$05      Tabellen zu den fünf Dateinamen löschen
C2FD[1] 9D 79 02  STA $0279,X   Endposition des Dateinamen im Puffer
C300    95 D7     STA $D7,X     Diectorysektor der Datei
C302    95 DC     STA $DC,X     Position der Datei in Directorysektor
```

```
C304  95 E1     STA $E1,X      Laufwerksnummer der Datei
C306  95 E6     STA $E6,X      Dateityp und Jokerflags
C308  9D 7F 02  STA $027F,X    aktuelle Spurnummer der Datei
C30B  9D 84 02  STA $0284,X    aktuelle Sektornummer der Datei
C30E  CA        DEX            Tabellen für nächsten Dateinamen
C30F  D0 EC     BNE $C2FD      bereits alle fünf möglichen Dateinamen ?
C311  60        RTS            zurück zur aufrufenden Routine
--------------------------------------------------------------------------
[90B8/A60E/D84C/EE0D/C320:C826,C90F,CA88,DA96]
Laufwerksnummern der Dateien holen und in Dateitabellen setzen
C312  AD 78 02  LDA $0278      Zahl der Dateien für die Datei-
C315  8D 77 02  STA $0277      nennung retten
C318  A9 01     LDA #$01       Anzahl der zu bearbeitenden
C31A  8D 78 02  STA $0278      auf eine Dateiangabe
C31D  8D 79 02  STA $0279      setzen
C320  AC 8E 02  LDY $028E      zuletzt aktives Laufwerk
C323  A2 00     LDX #$00       aktuelle Nummer des Dateinamens
C325  86 D3     STX $D3        Nummer merken
C327  BD 7A 02  LDA $027A,X    Startposition des Dateinamens im Puffer
C32A  20 3C C3  JSR $C33C      Laufwerksnummer aus Puffer holen
C32D  A6 D3     LDX $D3        Nummer des aktuellen Dateiparameters
C32F  9D 7A 02  STA $027A,X    Position im Befehlstring merken
C332  98        TYA            dazu Position des Laufwerks holen
C333  95 E2     STA $E2,X      Laufwerksnummer für Datei setzen
C335  E8        INX            nächste Datei anwählen
C336  EC 78 02  CPX $0278      mit Anzahl der zu bearbeitenden Dateien
C339  90 EA     BCC $C325      vergleichen; bereits alle fertig ?
C33B  60        RTS            ja, zurück zur aufrufenden Routine
--------------------------------------------------------------------------
[C32A]
Laufwerksnummer aus Befehlsstring holen
C33C  AA        TAX            Position des Dateinamens im Puffer
C33D  A0 00     LDY #$00       Nummer des Standardlaufwerks
C33F  A9 3A     LDA #$3A       Doppelpunkt ':'
C341  DD 01 02  CMP $0201,X    Doppelpunkt hinter aktueller Position ?
C344  F0 0C     BEQ $C352      ja, dann Syntax richtig und nach $C352
C346  DD 00 02  CMP $0200,X    mit aktueller Position vergleichen
C349  D0 01     BNE $C361      steht Zeiger auf Doppelpunkt ?
C34B  E8        INX            ja, dann keine Laufwerksangabe
C34C  98        TYA            Nummer des Laufwerks
C34D  29 01     AND #$01       nur 0 oder 1 erlauben
```

```
C34F[1] A8        TAY          und Laufwerk in Y-Register merken
C350 8A           TXA          aktuelle Position im Eingabepuffer
C351 60           RTS          zurück zur aufrufenden Routine
C352[1] BD 00 02  LDA $0200,X  Nummer des Laufwerks aus Befehlsstring
C355 E8           INX          Zeiger in Eingabepuffer hinter
C356 E8           INX          Laufwerksnennung (':') setzen
C357 C9 30        CMP #$30     Laufwerk Null ?
C359 F0 F2        BEQ $C34D    ja, dann Laufwerk setzen
C35B C9 31        CMP #$31     auf Laufwerk 1 prüfen
C35D F0 EE        BEQ $C34D    identisch ?
C35F D0 EB        BNE $C34C    nein, immer Sprung nach $C34C
C361[1] 98        TYA          Standard Laufwerksnummer (0)
C362 09 80        ORA #$80     Flag für mangelhafte
C364 29 81        AND #$81     Laufwerksnummer setzen
C366 D0 E7        BNE $C34F    und an aufrufende Routine übergeben
-----------------------------------------------------------------

[C1E2]
Initialisiert Laufwerk und schaltet LED am Laufwerk ein.
C368 A9 00        LDA #$00     Befehlssyntaxflag
C36A 8D 8B 02     STA $028B    löschen
C36D AC 7A 02     LDY $027A    aktuelle Position im Eingabepuffer
C370[1] B1 A3     LDA ($A3),Y  Zeichen aus Befehlsstring holen
C372 20 BD C3     JSR $C3BD    testet auf gültige Laufwerksnummer
C375 10 11        BPL $C388    Nummer richtig ?
C377 C8           INY          nein, Zeiger auf nächstes Zeichen
C378 CC 74 02     CPY $0274    Länge des Befehlsstrings
C37B B0 06        BCS $C383    Ende erreicht ?
C37D AC 74 02     LDY $0274    nein, Länge des Befehlsstrings
C380 88           DEY          Zeiger auf letztes Zeichen setzen
C381 D0 ED        BNE $C370    ist nur 1 Befehlsbuchstabe vorhanden ?
C383[1] CE 8B 02  DEC $028B    ja, 'nicht gefunden' in Syntaxflag
C386 A9 00        LDA #$00     Nummer des Standardlaufwerks
C388[1] 29 01     AND #$01     als aktuelle Laufwerksnnummer
C38A 85 7F        STA $7F      setzen
C38C 4C 00 C1     JMP $C100    LED einschalten
-----------------------------------------------------------------

[C40E/C420/C427/C467/C497/C704/C70B]
Auf anderes Laufwerk umschalten
C38F A5 7F        LDA $7F      aktuelle Laufwerksnummer
C391 49 01        EOR #$01     Bit für Laufwerk umdrehen und
C393 29 01        AND #$01     andere Bits ausblenden
```

```
C395  85 7F     STA $7F       als aktuelles Laufwerk speichern
C397  60        RTS           zurück zur aufrufenden Routine
-----------------------------------------------------------------
```

[C823/DA98]

Dateityp festellen und setzen

```
C398  A0 00     LDY #$00      ersten Dateinamen für Tabelle auswählen
C39A  AD 77 02  LDA $0277     Position des 1. Dateinamens
C39D  CD 78 02  CMP $0278     auf Position der Dateitypkennung prüfen
C3A0  F0 16     BEQ $C3B8     identisch ?
C3A2  CE 78 02  DEC $0278     nein, dann Zeiger auf Dateitypzeichen
C3A5  AC 78 02  LDY $0278     setzen
C3A8  B9 7A 02  LDA $027A,Y   Zeiger auf Endposition des Dateinamens
C3AB  A8        TAY           übernehmen und dazugehöriges
C3AC  B1 A3     LDA ($A3),Y   Zeichen aus dem Dateinamen holen
C3AE  A0 04     LDY #$04      Anzahl der möglichen Dateitypen
C3B0  D9 BB FE  CMP $FEBB,Y   Zeichen in der Tabelle der Dateitypen
C3B3  F0 03     BEQ $C3B8     vorhanden ?
C3B5  88        DEY           Zeiger auf nächsten Dateityp richten
C3B6  D0 F8     BNE $C3B0     bereits alle Dateitypen getestet
C3B8  98        TYA           ja, Nummer des Dateityps
C3B9  8D 96 02  STA $0296     (=0 wenn nicht vorhanden) merken
C3BC  60        RTS           zurück zur aufrufenden Routine
-----------------------------------------------------------------
```

[C372/DA68]

Laufwerksnummer auf Gültigkeit prüfen

```
C3BD  C9 30     CMP #$30      Ist Laufwerksnummer
C3BF  F0 06     BEQ $C3C7     gleich Laufwerk 0 ?
C3C1  C9 31     CMP #$31      nein, ist Laufwerksnummer dann
C3C3  F0 02     BEQ $C3C7     gleich Laufwerk 1 ?
C3C5  09 80     ORA #$80      nein, Bit7 als Fehlerflag setzen und
C3C7  29 81     AND #$81      restliche Bits (vom Ascii-Wert) löschen
C3C9  60        RTS           zurück zur aufrufenden Routine
-----------------------------------------------------------------
```

[C44F/C829/D84F/DA9E]

In Dateinamen angegebene Laufwerke initialisieren

```
C3CA  A9 00     LDA #$00      Zwischenspeicher zur Erzeugung
C3CC  85 6F     STA $6F       des Index auf Steuerbyte löschen
C3CE  8D 8D 02  STA $028D     Flag für Anzahl der Laufwerke löschen
C3D1  48        PHA           Stack für folgendes Programm präparieren
C3D2  AE 78 02  LDX $0278     Anzahl der zur Nennung gehörigen Dateien
C3D5  68        PLA           Zeiger auf Steuerbyte
```

```
C3D6     05 6F      ORA $6F        letztes Ergebnis eintragen
C3D8     48         PHA            und merken
C3D9     A9 01      LDA #$01       Flag für 'Laufwerksnennung vorhanden'
C3DB     85 6F      STA $6F        setzen
C3DD     CA         DEX            Zähler für zu prüfende Dateinamen
C3DE     30 0F      BMI $C3EF      bereits Laufwerksnummern aller Dateien ?
C3E0     B5 E2      LDA $E2,X      Laufwerksnummer der Datei holen
C3E2     10 04      BPL $C3E8      Laufwerksnennung gesetzt ?
C3E4     06 6F      ASL $6F        nein, Bitflags
C3E6     06 6F      ASL $6F        korregieren
C3E8[1]  4A         LSR A          Laufwerksnummer testen
C3E9     90 EA      BCC $C3D5      ist Laufwerk 1 angewählt ?
C3EB     06 6F      ASL $6F        ja, Zeiger auf Bytes für Laufwerk 1
C3ED     D0 E6      BNE $C3D5      immer Sprung nach $C3D5
C3EF[1]  68         PLA            Zeiger auf Steuerbyte
C3F0     AA         TAX            für Laufwerksinitialisierung setzen
C3F1     BD 3F C4   LDA $C43F,X    Steuerbyte für Zugriff holen
C3F4     48         PHA            und merken
C3F5     29 03      AND #$03       Zahl der erlaubten Laufwerke
C3F7     8D 8C 02   STA $028C      feststellen und merken
C3FA     68         PLA            Steuerbyte wieder holen
C3FB     0A         ASL A          Flag für 'nur eine Laufwerksnennung'
C3FC     10 3E      BPL $C43C      ist nur eine Nennung erlaubt ?
C3FE     A5 E2      LDA $E2        ja, Laufwerksnummer der ersten Datei
C400     29 01      AND #$01       als aktuelles Laufwerk
C402     85 7F      STA $7F        übernehmen
C404     AD 8C 02   LDA $028C      Zahl der erlaubten Laufwerke holen
C407     F0 2B      BEQ $C434      nur ein Laufwerk zugelassen ?
C409     20 3D C6   JSR $C63D      nein, aktuelles Laufwerk initialisieren
C40C     F0 12      BEQ $C420      Laufwerk bereit ?
C40E     20 8F C3   JSR $C38F      nein, auf anderes Laufwerk umschalten
C411     A9 00      LDA #$00       Zahl der erlaubten Laufwerke
C413     8D 8C 02   STA $028C      löschen
C416     20 3D C6   JSR $C63D      anderes Laufwerk initialisieren
C419     F0 1E      BEQ $C439      Laufwerk bereit ?
C41B[1]  A9 74      LDA #$74       nein, Fehlermeldung
C41D     20 C8 C1   JSR $C1C8      '74 Drive Not Ready' ausgeben
C420[1]  20 8F C3   JSR $C38F      auf anderes Laufwerk wechseln
C423     20 3D C6   JSR $C63D      Laufwerk initialisieren
C426     08         PHP            und Ergebnis merken
C427     20 8F C3   JSR $C38F      auf anderes Laufwerk wechseln und
```

```
C42A    28        PLP         vorheriges Ergebnis wieder holen
C42B    F0 0C     BEQ $C439   ist vorheriges Laufwerk bereit ?
C42D    A9 00     LDA #$00    nein, Zahl der erlaubten Laufwerke
C42F    8D 8C 02  STA $028C   löschen
C432    F0 05     BEQ $C439   immer Sprung nach $C439
C434[1] 20 3D C6  JSR $C63D   Laufwerk initialisieren
C437    D0 E2     BNE $C41B   ist Laufwerk bereit ?
C439[3] 4C 00 C1  JMP $C100   ja, LED am Laufwerk einschalten
C43C[1] 2A        ROL A       Steuerbyte korregieren und
C43D    4C 00 C4  JMP $C400   Laufwerksnummer aus Steuerbyte holen
-----------------------------------------------------------------------
[C3F1]  Steuerbytes für Art der Laufwerksinitialisierung

Funktion der einzelnen Bits :
Bit 0/1 : Zahl der anzusprechenden Laufwerke (0/1/2)
Bit 6   : 1= Laufwerksnummer aus Steuerbyte übernehmen
Bit 7   : 0/1 Laufwerksnummer für Bit6

C440  00 80 41 01 01 01 01 81
C448  81 81 81 42 42 42 42
-----------------------------------------------------------------------
[90C8/C952/CA99/E7B8]
Dateieintrag in Directory suchen
C44F    20 CA C3  JSR $C3CA   Laufwerk für zu suchende Datei setzen
C452[1] A9 00     LDA #$00    Zeiger auf ersten Directoryeintrag
C454    8D 92 02  STA $0292   löschen
C457    20 AC C5  JSR $C5AC   Zeiger setzen. Eintrag suchen
C45A    D0 19     BNE $C475   gültiger Eintrag gefunden ?
C45C[1] CE 8C 02  DEC $028C   nein, Zahl der Laufwerke
C45F    10 01     BPL $C462   noch ein Laufwerk ?
C461    60        RTS         zurück zur aufrufenden Routine
C462[1] A9 01     LDA #$01    Flag für 'auf beiden Laufwerken
C464    8D 8D 02  STA $028D   suchen' setzen
C467    20 8F C3  JSR $C38F   auf anderes Laufwerk wechseln
C46A    20 00 C1  JSR $C100   LED am aktuellen Laufwerk anschalten
C46D    4C 52 C4  JMP $C452   Eintrag auf anderem Laufwerk suchen
C470[1] 20 17 C6  JSR $C617   nächste gültige Datei suchen
C473    F0 10     BEQ $C485   gefunden ?
C475[2] 20 D8 C4  JSR $C4D8   ja, Directoryeintrag überprüfen
C478    AD 8F 02  LDA $028F   Zeiger für Dateieintrag gefunden
C47B    F0 01     BEQ $C47E   Eintrag richtig ?
```

```
C47D    60        RTS           ja, zurück zur aufrufenden Routine
C47E[1] AD 53 02  LDA $0253     Flag für Eintrag gefunden
C481    30 ED     BMI $C470     Datei gefunden (A <> $FF) ?
C483    10 F0     BPL $C475     ja, immer Sprung nach $C475
C485[1] AD 8F 02  LDA $028F     Flag für Datei gefunden
C488    F0 D2     BEQ $C45C     letzter Eintrag ?
C48A    60        RTS           nein, zurück zur aufrufenden Routine
-----------------------------------------------------------------------
[C86D/C49D:C830,D852,DAA4/C4B5:C7A7]
Dateieintrag im Directory suchen
C48B    20 04 C6  JSR $C604     Directory nach Datei durchsuchen
C48E    F0 1A     BEQ $C4AA     Eintrag gefunden ?
C490    D0 28     BNE $C4BA     ja, weiter bei $C4BA
C492[1] A9 01     LDA #$01      Flag für Zugriff auf beide Laufwerke
C494    8D 8D 02  STA $028D     setzen
C497    20 8F C3  JSR $C38F     auf anderes Laufwerk wechseln
C49A    20 00 C1  JSR $C100     LED einschalten
C49D[3] A9 00     LDA #$00      Zeiger auf ersten gültigen
C49F    8D 92 02  STA $0292     Eintrag löschen
C4A2    20 AC C5  JSR $C5AC     Zeiger initialisieren; Eintrag suchen
C4A5    D0 13     BNE $C4BA     Datei gefunden ?
C4A7    8D 8F 02  STA $028F     Position
C4AA[2] AD 8F 02  LDA $028F     merken
C4AD    D0 28     BNE $C4D7     letzter Eintrag
C4AF    CE 8C 02  DEC $028C     ja, Zahl der erlaubten Laufwerke
C4B2    10 DE     BPL $C492     auf anderes Laufwerk umschalten ?
C4B4    60        RTS           nein, zurück zur aufrufenden Routine
C4B5[3] 20 17 C6  JSR $C617     nächsten Eintrag holen
C4B8    F0 F0     BEQ $C4AA     Eintrag gefunden ?
C4BA[2] 20 D8 C4  JSR $C4D8     ja, Eintrag mit Gesuchtem vergleichen
C4BD    AE 53 02  LDX $0253     Flag holen
C4C0    10 07     BPL $C4C9     Eintrag identisch ?
C4C2    AD 8F 02  LDA $028F     nein, Flag für Datei gefunden holen
C4C5    F0 EE     BEQ $C4B5     wurde Datei gefunden ?
C4C7    D0 0E     BNE $C4D7     nein, immer Sprung nach $C4D7
C4C9[1] AD 96 02  LDA $0296     aktuellen Dateityp holen
C4CC    F0 09     BEQ $C4D7     ist Eintrag gültig ?
C4CE    B5 E7     LDA $E7,X     ja, Zeiger für gesuchten Dateityp holen
C4D0    29 07     AND #$07      und Kennzeichen für Typ isolieren
C4D2    CD 96 02  CMP $0296     mit gesuchtem Dateityp vergleichen
C4D5    D0 DE     BNE $C4B5     identisch ?
```

```
C4D7[3] 60          RTS            ja, zurück zur aufrufenden Routine
--------------------------------------------------------------------------
[C475/C4BA]
Directoryeintrag suchen
C4D8   A2 FF       LDX #$FF       Flag für 'Eintrag gefunden'
C4DA   8E 53 02    STX $0253      löschen
C4DD   E8          INX            (0)
C4DE   8E 8A 02    STX $028A      Flag für 'Joker vorhanden' löschen
C4E1   20 89 C5    JSR $C589      Dateiflag setzen
C4E4   F0 06       BEQ $C4EC      wurde Eintrag gefunden ?
C4E6[1] 60          RTS            ja, zurück zur aufrufenden Routine
--------------------------------------------------------------------------
[C4F5/C4FC/C513/C519/C533]
nächsten Eintrag suchen
C4E7   20 94 C5    JSR $C594      nächsten Eintrag suchen
C4EA   D0 FA       BNE $C4E6      gefunden ?
C4EC[1] A5 7F       LDA $7F        ja, aktuelles Laufwerk holen
C4EE   55 E2       EOR $E2,X      und mit Laufwerksnummer der Dateiangabe
C4F0   4A          LSR A          vergleichen
C4F1   90 0B       BCC $C4FE      identisch ?
C4F3   29 40       AND #$40       nein, Flag für Laufwerkstyp holen
C4F5   F0 F0       BEQ $C4E7      Laufwerk durch Standardwert gesetzt ?
C4F7   A9 02       LDA #$02       ja, Wert für Zugriff auf beide Laufwerke
C4F9   CD 8C 02    CMP $028C      mit Zugriffsflag vergleichen
C4FC   F0 E9       BEQ $C4E7      auf beiden Laufwerken suchen ?
C4FE[1] BD 7A 02    LDA $027A,X    nein, Position des Dateinamens im
C501   AA          TAX            Befehlsstring holen und merken
C502   20 A6 C6    JSR $C6A6      Parameter für Name im Befehl setzen
C505   A0 03       LDY #$03       Pufferzeiger auf Directoryname setzen
C507   4C 1D C5    JMP $C51D      Namen mit Befehlsstring vergleichen
C50A[1] BD 00 02    LDA $0200,X    Zeichen aus Befehlsstring holen und so
C50D   D1 94       CMP ($94),Y    Name mit Directoryname vergleichen
C50F   F0 0A       BEQ $C51B      identisch ?
C511   C9 3F       CMP #$3F       nein, mit '?' Joker vergleichen
C513   D0 D2       BNE $C4E7      identisch ?
C515   B1 94       LDA ($94),Y    ja, Zeichen aus Directoryeintrag holen
C517   C9 A0       CMP #$A0       mit Wert für 'Shift Space' vergleichen
C519   F0 CC       BEQ $C4E7      schon ganzer Dateiname gelesen ?
C51B[1] E8          INX            nein, Zeiger auf Befehlsstring
C51C   C8          INY            Zeiger in Directorypuffer richten
C51D[1] EC 76 02    CPX $0276      Zeiger auf Ende des Namens im Befehl
```

```
C520     B0 09      BCS $C52B      Ende des Dateinamens erreicht ?
C522     BD 00 02   LDA $0200,X    nein, Zeichen aus Dateiname holen
C525     C9 2A      CMP #$2A       mit Zeichen für '*' Joker vergleichen
C527     F0 0C      BEQ $C535      identisch ?
C529     D0 DF      BNE $C50A      nein, immer Sprung mach $C50A
C52B[1]  C0 13      CPY #$13       mit 'Return' vergleichen
C52D     B0 06      BCS $C535      ist ASCII-Wert kleiner ?
C52F     B1 94      LDA ($94),Y    ja, Zeichen aus Directory holen und
C531     C9 A0      CMP #$A0       mit Wert für 'Shift Space' vergleichen
C533     D0 B2      BNE $C4E7      schon ganzer Dateiname gelesen ?
C535[2]  AE 79 02   LDX $0279      ja, Position des Directoryeintrags holen
C538     8E 53 02   STX $0253      und in Zeiger setzen
C53B     B5 E7      LDA $E7,X      Dateityp des Eintrags feststellen
C53D     29 80      AND #$80       Flag für Joker setzen
C53F     8D 8A 02   STA $028A      und merken
C542     AD 94 02   LDA $0294      Zeiger auf Position im Directorypuffer
C545     95 DD      STA $DD,X      in Tabelle dem Dateinamen zuordnen
C547     A5 81      LDA $81        Nummer des Directorysektors
C549     95 D8      STA $D8,X      merken
C54B     A0 00      LDY #$00       Pufferzeiger auf Start des Eintrags
C54D     B1 94      LDA ($94),Y    Dateityp aus Directory holen
C54F     C8         INY            Pufferzeiger auf nächstes Zeichen
C550     48         PHA            Orginal-Dateityp merken
C551     29 40      AND #$40       Flag für Scratch-Schutz isolieren
C553     85 6F      STA $6F        und merken
C555     68         PLA            Dateityp wieder holen
C556     29 DF      AND #$DF       und Scratch-Flag ausblenden
C558     30 02      BMI $C55C      ist Datei ordnungsgemäß geschlossen ?
C55A     09 20      ORA #$20       nein, Flag für '*' Datei
C55C[1]  29 27      AND #$27       Flag und Dateitypkennzeichen übernehmen
C55E     05 6F      ORA $6F        und Scratch-Flag einblenden
C560     85 6F      STA $6F        beides merken
C562     A9 80      LDA #$80       Flag für 'Dateityp gesetzt'
C564     35 E7      AND $E7,X      aus Tabelle übertragen
C566     05 6F      ORA $6F        und Bits des neuen Dateitypen einblenden
C568     95 E7      STA $E7,X      Typ in Tabelle dem Dateinamen zuordnen
C56A     B5 E2      LDA $E2,X      Nummer des Laufwerks des Eintrags holen
C56C     29 80      AND #$80       und aktuelle Laufwerksnummer
C56E     05 7F      ORA $7F        eintragen
C570     95 E2      STA $E2,X      Wert in Laufwerkstabelle schreiben
C572     B1 94      LDA ($94),Y    Spurnummer des ersten Sektors holen
```

```
C574   9D 80 02   STA $0280,X      und in Tabelle eintragen
C577   C8         INY              Zeiger auf nächstes Byte richten
C578   B1 94      LDA ($94),Y      Sektornummer aus Eintrag holen
C57A   9D 85 02   STA $0285,X      und merken
C57D   AD 58 02   LDA $0258        aktuelle Recordlänge holen
C580   D0 07      BNE $C589        ist Wert gesetzt ?
C582   A0 15      LDY #$15         nein, Pufferzeiger auf Wert in Eintrag
C584   B1 94      LDA ($94),Y      Recordlänge des Directoryeintrags holen
C586   8D 58 02   STA $0258        und in Zeiger merken
-----------------------------------------------------------------------
[C4E1/C580]
Flags neu initialisieren
C589   A9 FF      LDA #$FF         Zeiger löschen
C58B   8D 8F 02   STA $028F        Flag für letzten Eintrag
C58E   AD 78 02   LDA $0278        Zeiger auf Position des Dateinamens
C591   8D 79 02   STA $0279        im Eingabepuffer
-----------------------------------------------------------------------
[C4E7/C5A4]
Dateinamen-Suche abschließen
C594   CE 79 02   DEC $0279        Zahl der Dateinamen
C597   10 01      BPL $C59A        noch einen Eintrag bearbeiten ?
C599   60         RTS              nein, zurück zur aufrufenden Routine
C59A[1] AE 79 02  LDX $0279        Nummer des Dateinamens holen
C59D   B5 E7      LDA $E7,X        und dazugehörigen Dateityp feststellen
C59F   30 05      BMI $C5A6        ist Wert gesetzt ?
C5A1   BD 80 02   LDA $0280,X      nein, Spur des ersten Sektors holen
C5A4   D0 EE      BNE $C594        ist Wert angegeben ?
C5A6[1] A9 00     LDA #$00         nein, Flag für 'letzten Directoryeintrag
C5A8   8D 8F 02   STA $028F        erreicht' setzen
C5AB   60         RTS              zurück zur aufrufenden Routine
-----------------------------------------------------------------------
[C457/C4A2/D70E/ED97/C604:C48B,EDD4]
Zeiger für Suche im Directory setzen
C5AC   A0 00      LDY #$00         Zeiger auf aktuellen Directorysektor
C5AE   8C 91 02   STY $0291        löschen
C5B1   88         DEY              Flag für 'Eintrag gefunden'
C5B2   8C 53 02   STY $0253        zurücksetzen
C5B5   AD 85 FE   LDA $FE85        Nummer der Directoryspur (18)
C5B8   85 80      STA $80          holen und als aktuelle Spur merken
C5BA   A9 01      LDA #$01         Zeiger auf Sektornummer
C5BC   85 81      STA $81          setzen
```

```
C5BE   8D 93 02   STA $0293     Flag für 'Sektor gelesen' löschen
C5C1   20 75 D4   JSR $D475     Sektor in Puffer einlesen
C5C4[1] AD 93 02  LDA $0293     Zeiger auf nächsten Directorysektor
C5C7   D0 01      BNE $C5CA     gibt es einen Folgesektor ?
C5C9   60         RTS           nein, ja, zurück zur aufrufenden Routine
C5CA[1] A9 07     LDA #$07      Zahl der Dateieintrage in einem
C5CC   8D 95 02   STA $0295     Directoryblock -1 merken
C5CF   A9 00      LDA #$00      Position des zu lesenden Bytes
C5D1   20 F6 D4   JSR $D4F6     Byte aus dem aktuellen Puffer holen
C5D4   8D 93 02   STA $0293     und merken
C5D7[1] 20 E8 D4  JSR $D4E8     Pufferzeiger für aktuellen Puffer setzen
C5DA   CE 95 02   DEC $0295     Zähler für Einträge pro Directorysektor
C5DD   A0 00      LDY #$00      Zeiger auf Beginn des Eintrags setzen
C5DF   B1 94      LDA ($94),Y   und Dateitypkennzeichen holen
C5E1   D0 18      BNE $C5FB     ist Eintrag gelöscht ?
C5E3   AD 91 02   LDA $0291     ja, Nummer des aktuellen Sektors
C5E6   D0 2F      BNE $C617     ist Wert gesetzt ?
C5E8   20 3B DE   JSR $DE3B     nein, Spur und Sektornummer holen
C5EB[1] A5 81     LDA $81       Nummer des aktuellen Directorysektors
C5ED   8D 91 02   STA $0291     merken
C5F0   A5 94      LDA $94       Low-Byte des Zeigers auf Eintrag
C5F2   AE 92 02   LDX $0292     Zeiger auf gültigen Eintrag holen
C5F5   8D 92 02   STA $0292     neuen Wert setzen
C5F8   F0 1D      BEQ $C617     war Zeiger vorher gelöscht ?
C5FA   60         RTS           nein, zurück zur aufrufenden Routine
C5FB*  A2 01      LDX #$01      Nummer des ersten Eintrags
C5FD   EC 92 02   CPX $0292     mit letztem Wert vergleichen
C600   D0 2D      BNE $C62F     war erster Einntrag gesetzt ?
C602   F0 13      BEQ $C617     nein, immer Sprung nach $C617
C604[2] AD 85 FE  LDA $FE85     Nummer der Directoryspur
C607   85 80      STA $80       holen und als aktuelle Spur merken
C609   AD 90 02   LDA $0290     Nummer des Directorysektors
C60C   85 81      STA $81       als aktuellen Sektor vermerken
C60E   20 75 D4   JSR $D475     Sektor von Diskette in Puffer lesen
C611   AD 94 02   LDA $0294     Zeiger auf Position des Eintrags
C614   20 C8 D4   JSR $D4C8     Directoryzeiger setzen
C617[6] A9 FF     LDA #$FF      Flag für 'Dateieintrag gefunden'
C619   8D 53 02   STA $0253     löschen
C61C   AD 95 02   LDA $0295     Zahl der Directoryeinträge pro Sektor
C61F   30 08      BMI $C629     ist Zähler gesetzt ?
C621   A9 20      LDA #$20      ja, Zahl der Bytes eines Eintrags
```

```
C623    20 C6 D1   JSR $D1C6      Pufferzeiger auf nächsten Dateieintrag
C626    4C D7 C5   JMP $C5D7      Zeiger setzen
C629[1] 20 4D D4   JSR $D44D      nächsten Block des Directory lesen
C62C    4C C4 C5   JMP $C5C4      Zeiger setzen
C62F[1] A5 94      LDA $94        Low-Byte des aktuellen Zeigers
C631    8D 94 02   STA $0294      merken
C634    20 3B DE   JSR $DE3B      Spur und Sektor des letzten Jobs holen
C637    A5 81      LDA $81        Nummer des Directorysektors
C639    8D 90 02   STA $0290      merken
C63C    60         RTS            zurück zur aufrufenden Routine
-----------------------------------------------------------------------

[83A0/840B/9088/C409/C416/C423/C434/CB8C/C664:A9A9]
Diskette initialisieren
C63D    A5 68      LDA $68        Flag für 'autom. Initialisieren'
C63F    D0 28      BNE $C669      nur initialisieren von Hand erlaubt ?
C641    A6 7F      LDX $7F        nein, Nummer des aktuellen Laufwerks
C643    56 1C      LSR $1C,X      holen und Flag für Initialisieren testen
C645    90 22      BCC $C669      soll Diskette initialisiert werden ?
C647    A9 FF      LDA #$FF       ja, Flag für 'Fehler bei Job beachten'
C649    8D 98 02   STA $0298      löschen
C64C    20 0E D0   JSR $D00E      Prüfen ob eine Disk im Laufwerk liegt
C64F    A0 FF      LDY #$FF       Flagwert für 'Fehler aufgetreten'
C651    C9 02      CMP #$02       Ergebnis mit Kode für Sync vergleichen
C653    F0 0A      BEQ $C65F      wurde Sync-Markierung gefunden ?
C655    C9 03      CMP #$03       ja, auf Kode für Blockheader prüfen
C657    F0 06      BEQ $C65F      ist ein Blockheader gefunden worden ?
C659    C9 0F      CMP #$0F       ja, auf Kode für Laufwerk aktiv testen
C65B    F0 02      BEQ $C65F      ist das Laufwerk ansprechbar ?
C65D    A0 00      LDY #$00       ja, Flagwert für 'kein Fehler'
C65F[3] A6 7F      LDX $7F        Nummer des aktuellen Laufwerks holen
C661    98         TYA            und das Fehlerflag
C662    95 FF      STA $FF,X      in entsprechenden Laufwerksstatus
C664[1] D0 03      BNE $C669      ist Laufwerk bereit ?
C666    20 42 D0   JSR $D042      ja, BAM einlesen
C669[3] A6 7F      LDX $7F        Nummer des aktuellen Laufwerks und
C66B    B5 FF      LDA $FF,X      dazugehörigen Laufwerksstatus holen
C66D    60         RTS            zurück zur aufrufenden Routine
-----------------------------------------------------------------------
```

```
[CAC0/D768/EE68]
Dateiname aus Eingabepuffer in Directorypuffer kopieren
(Im Akku muß die Länge des Namens, in X die Position im Eingabestring und
in Y die Nummer des Directorypuffers stehen)
C66E   48         PHA            Länge des Dateinamens merken
C66F   20 A6 C6   JSR $C6A6      Position des Namens im Eingabestring
C672   20 88 C6   JSR $C688      feststellen und Namen in Puffer kopieren
C675   68         PLA            Länge des Dateinamens wieder holen
C676   38         SEC            Länge des kopierten Dateinamens
C677   ED 4B 02   SBC $024B      mit der maximalen Länge eines
C67A   AA         TAX            Dateieintrags (16) vergleichen
C67B   F0 0A      BEQ $C687      wird Eintrag voll ausgefüllt ?
C67D   90 08      BCC $C687      nein, ist Dateiname kleiner als Platz ?
C67F   A9 A0      LDA #$A0       ja, dann restliche Zeichen
C681[1] 91 94     STA ($94),Y    des Dateinamens mit 'Shift Space' füllen
C683   C8         INY            Pufferzeiger auf nächste Zeichenposition
C684   CA         DEX            Zahl der aufzufüllenden Zeichen
C685   D0 FA      BNE $C681      Dateiname aufgefüllt ?
C687[2] 60        RTS            ja, zurück zur aufrufenden Routine
------------------------------------------------------------------------
[C672]
Kopiert einen Teil aus dem Eingabepuffer in den aktuellen Datenpuffer
C688   98         TYA            Nummer des aktuellen
C689   0A         ASL A          Puffers verdoppeln
C68A   A8         TAY            (da Tabelle 2-Byte Zeiger enthält)
C68B   B9 99 00   LDA $0099,Y    Adresse des Puffers (Low-Byte) holen
C68E   85 94      STA $94        und in Zeiger auf Directorypuffer setzen
C690   B9 9A 00   LDA $009A,Y    Adresse des Puffers (High-Byte) holen
C693   85 95      STA $95        und in Zeiger auf Directorypuffer setzen
C695   A0 00      LDY #$00       Zeiger auf Position in Puffer löschen
C697[1] BD 00 02  LDA $0200,X    Byte aus Eingabepuffer holen
C69A   91 94      STA ($94),Y    und in aktuellen Puffer kopieren
C69C   C8         INY            Zeiger in Datenpuffer auf nächstes Byte
C69D   F0 06      BEQ $C6A5      Datenpuffer voll ?
C69F   E8         INX            nein, Zeiger in Eingabepuffer erhöhen
C6A0   EC 76 02   CPX $0276      mit Länge des Befehlsstrings vergleichen
C6A3   90 F2      BCC $C697      letztes Zeichen erreicht ?
C6A5[1] 60        RTS            ja, zurück zur aufrufenden Routine
------------------------------------------------------------------------
```

```
[C502/C66F]
Ende eines Dateinamens im Eingabepuffer suchen (Startposition in X)
C6A6   A9 00      LDA #$00       Zeiger auf Länge des Namens
C6A8   8D 4B 02   STA $024B      löschen
C6AB   8A         TXA            Startposition im Eingabepuffer holen
C6AC   48         PHA            und merken
C6AD[1] BD 00 02  LDA $0200,X    Zeichen des Namens holen
C6B0   C9 2C      CMP #$2C       und mit ',' vergleichen
C6B2   F0 14      BEQ $C6C8      identisch ?
C6B4   C9 3D      CMP #$3D       nein, mit '=' vergleichen
C6B6   F0 10      BEQ $C6C8      identisch ?
C6B8   EE 4B 02   INC $024B      nein, Länge des Dateinamens erhöhen
C6BB   E8         INX            Pufferzeiger auf nächstes Zeichen setzen
C6BC   A9 0F      LDA #$0F       maximale Länge eines Dateinamens mit
C6BE   CD 4B 02   CMP $024B      Länge des aktuellen Namens vergleichen
C6C1   90 05      BCC $C6C8      aktueller Dateiname zu groß ?
C6C3   EC 74 02   CPX $0274      nein, Position m. Stringende vergleichen
C6C6   90 E5      BCC $C6AD      Ende des Eingabestrings erreicht ?
C6C8[3] 8E 76 02  STX $0276      ja, Länge des Dateinamens merken
C6CB   68         PLA            u. Zeiger auf Startposition wieder holen
C6CC   AA         TAX            und setzen
C6CD   60         RTS            zurück zur aufrufenden Routine
---------------------------------------------------------------------
[ECF2]
Dateieintrag aus dem Directory lesen
C6CE   A5 83      LDA $83        aktuell aktive Sekundäradresse
C6D0   48         PHA            retten
C6D1   A5 82      LDA $82        Nummer des aktuell aktiven Kanals
C6D3   48         PHA            retten
C6D4   20 DE C6   JSR $C6DE      Dateieintrag holen
C6D7   68         PLA            Nummer des Kanals
C6D8   85 82      STA $82        wieder herstellen
C6DA   68         PLA            und vorherige Sekundäradresse
C6DB   85 83      STA $83        wieder setzen
C6DD   60         RTS            zurück zur aufrufenden Routine
---------------------------------------------------------------------
[C6D4]
Directory für Ausgabe in Puffer erstellen
C6DE   A9 11      LDA #$11       Sekundäradresse 17
C6E0   85 83      STA $83        festlegen
C6E2   20 EB D0   JSR $D0EB      Kanal öffnen
```

```
C6E5    20 E8 D4   JSR $D4E8     Zeiger auf aktuellen Puffer setzen
C6E8    AD 53 02   LDA $0253     Flag für Dateieintrag holen
C6EB    10 0A      BPL $C6F7     wurde Eintrag gefunden ?
C6ED    AD 8D 02   LDA $028D     nein, Flag für Directorylaufwerke
C6F0    D0 0A      BNE $C6FC     Dirctory von beiden Laufwerken ?
C6F2    20 06 C8   JSR $C806     nein, 'Blocks Free' holen und in Puffer
C6F5    18         CLC           schreiben
C6F6    60         RTS           zurück zur aufrufenden Routine
C6F7[1] AD 8D 02   LDA $028D     Flag für Directorylaufwerke testen
C6FA    F0 1F      BEQ $C71B     Directory von beiden Laufwerken ?
C6FC[1] CE 8D 02   DEC $028D     ja, Flag auf 'nein' setzen
C6FF    D0 0D      BNE $C70E     war Flag nicht richtig gesetzt ?
C701    CE 8D 02   DEC $028D     nein, Flag korregieren
C704    20 8F C3   JSR $C38F     auf anderes Laufwerk wechseln
C707    20 06 C8   JSR $C806     'Blocks Free' holen und in Puffer
C70A    38         SEC           schreiben und wieder auf
C70B    4C 8F C3   JMP $C38F     ursprüngliches Laufwerk schalten
C70E[1] A9 00      LDA #$00      Zwischenspeicher für Blockzahl
C710    8D 73 02   STA $0273     löschen
C713    8D 8D 02   STA $028D     Flag für 'beide Laufwerke' löschen
C716    20 B7 C7   JSR $C7B7     Titel des Directory erstellen
C719    38         SEC           Flag für 'noch weitere Einträge'
C71A    60         RTS           zurück zur aufrufenden Routine
C71B[1] A2 18      LDX #$18      Länge einer Directoryzeile (24)
C71D    A0 1D      LDY #$1D      Zeiger auf Byte für Dateilänge setzen
C71F    B1 94      LDA ($94),Y   Zahl der Blocks (High-Byte) holen
C721    8D 73 02   STA $0273     und merken
C724    F0 02      BEQ $C728     ist Blockzahl > 256 und so dreistellig ?
C726    A2 16      LDX #$16      ja, Länge der um zwei Zeichen kürzen
C728[1] 88         DEY           Pufferzeiger auf Byte für Blockzahl
C729    B1 94      LDA ($94),Y   Low-Byte der Blockzahl holen
C72B    8D 72 02   STA $0272     und merken
C72E    E0 16      CPX #$16      mit Wert für verkürzte Länge vergleichen
C730    F0 0A      BEQ $C73C     ist 3-stellige Blockzahl vorhanden ?
C732    C9 0A      CMP #$0A      Zahl der Blocks mit 10 vergleichen
C734    90 06      BCC $C73C     Blockzahl kleiner (einstellig) ?
C736    CA         DEX           nein, Länge der Restzeile verkürzen
C737    C9 64      CMP #$64      Blockzahl mit 100 vergleichen
C739    90 01      BCC $C73C     ist die Blockzahl kleiner (2-stellig) ?
C73B    CA         DEX           nein, Länge der Restzeile verkürzen
C73C[3] 20 AC C7   JSR $C7AC     Puffer für Directory löschen
```

```
C73F  B1 94     LDA ($94),Y    Byte für Dateityp holen
C741  48        PHA            und merken
C742  0A        ASL A          Bit 6 als Flag für Scratchschutz holen
C743  10 05     BPL $C74A      ist Datei Scratch geschützt ?
C745  A9 3C     LDA #$3C       ja, Zeichen für Scratchschutz '<'
C747  9D B2 02  STA $02B2,X    hinter Dateityp schreiben
C74A  68        PLA            Dateityp zurückholen
C74B  29 0F     AND #$0F       Dateiart isolieren
C74D  A8        TAY            und dazugehörigen Kurznamen in Directory
C74E  B9 C5 FE  LDA $FEC5,Y    3. Buchstabe der Dateikurzbezeichnung
C751  9D B1 02  STA $02B1,X    holen und in Puffer schreiben
C754  CA        DEX            Länge der Directoryzeile verkürzen
C755  B9 C0 FE  LDA $FEC0,Y    2. Buchstabe der Dateikurzbezeichnung
C758  9D B1 02  STA $02B1,X    holen und in Puffer schreiben
C75B  CA        DEX            Länge der Directoryzeile verkürzen
C75C  B9 BB FE  LDA $FEBB,Y    1. Buchstabe der Dateikurzbezeichnung
C75F  9D B1 02  STA $02B1,X    holen und in Puffer schreiben
C762  CA        DEX            Länge der Directoryzeile
C763  CA        DEX            verkürzen
C764  B0 05     BCS $C76B      ist die Datei ordungsgemäß geschlossen ?
C766  A9 2A     LDA #$2A       nein, '*' als Kennzeichen
C768  9D B2 02  STA $02B2,X    vor die Dateikurzbezeichnung setzen
C76B  A9 A0     LDA #$A0       ein Leerzeichen
C76D  9D B1 02  STA $02B1,X    einfügen und
C770  CA        DEX            Länge der Directoryzeile verkürzen
C771  A0 12     LDY #$12       Pufferposition des Dateinamens setzen
C773  B1 94     LDA ($94),Y    Zeichen des Dateinamens holen
C775  9D B1 02  STA $02B1,X    und in Directorypuffer schreiben
C778  CA        DEX            Länge der Directoryzeile verkürzen
C779  88        DEY            Pufferzeiger erniedrigen
C77A  C0 03     CPY #$03       mit Endwert vergleichen
C77C  B0 F5     BCS $C773      alle Zeichen des Namens übernommen ?
C77E  A9 22     LDA #$22       ja, Anführungszeichen vor Namen
C780  9D B1 02  STA $02B1,X    setzen
C783  E8        INX            Zeiger in Directoryzeile erhöhen
C784  E0 20     CPX #$20       auf maximalen Wert prüfen
C786  B0 0B     BCS $C793      Ende des Puffers erreicht ?
C788  BD B1 02  LDA $02B1,X    nein, Zeichen aus Dateinamen holen
C78B  C9 22     CMP #$22       und mit Anführungszeichen vergleichen
C78D  F0 04     BEQ $C793      identisch ?
C78F  C9 A0     CMP #$A0       nein, mit 'Shift Space' vergleichen
```

```
C791    D0 F0      BNE $C783       identisch ?
C793[2] A9 22      LDA #$22        ja, dann durch Anführungszeichen
C795    9D B1 02   STA $02B1,X     (am Ende des Dateinamens) ersetzen
C798[1] E8         INX             Zeiger auf Dateinamen auf nächstes Byte
C799    E0 20      CPX #$20        setzen und mit Endwert vergleichen
C79B    B0 0A      BCS $C7A7       Ende des Dateinamens erreicht ?
C79D    A9 7F      LDA #$7F        nein, Wert für Bit 7 (Revers) gelöscht
C79F    3D B1 02   AND $02B1,X     Zeichen der Directoryzeile holen
C7A2    9D B1 02   STA $02B1,X     und Reversdarstellung abschalten
C7A5    10 F1      BPL $C798       immer Sprung nach $C798
C7A7[1] 20 B5 C4   JSR $C4B5       nächsten Eintrag holen
C7AA    38         SEC             Flag für 'noch weitere Einträge'
C7AB    60         RTS             zurück zur aufrufenden Routine
-----------------------------------------------------------------
[C73C/C7BD/C806]
Puffer für Dateiname mit Leerzeichen löschen
C7AC    A0 1B      LDY #$1B        Länge der Directoryzeile (27)
C7AE    A9 20      LDA #$20        Leerzeichen als Löschwert
C7B0[1] 99 B0 02   STA $02B0,Y     Pufferposition löschen
C7B3    88         DEY             Pufferzeiger auf nächstes Byte setzen
C7B4    D0 FA      BNE $C7B0       schon ganzer Puffer gelöscht ?
C7B6    60         RTS             ja, zurück zur aufrufenden Routine
-----------------------------------------------------------------
[C716/DAA1]
Titel des Directory mit Namen und ID erzeugen
C7B7    20 19 F1   JSR $F119       Zeiger auf BAM setzen
C7BA    20 DF F0   JSR $F0DF       BAM von Diskette lesen
C7BD    20 AC C7   JSR $C7AC       Puffer für Directoryzeile löschen
C7C0    A9 FF      LDA #$FF        Zwischenspeicher
C7C2    85 6F      STA $6F         initialisieren
C7C4    A6 7F      LDX $7F         Nummer des aktuellen Laufwerks
C7C6    8E 72 02   STX $0272       als Zweibytewert (wie Blockzahl)
C7C9    A9 00      LDA #$00        in Directorypuffer
C7CB    8D 73 02   STA $0273       schreiben
C7CE    A6 F9      LDX $F9         Nummer des aktuellen Puffers
C7D0    BD E0 FE   LDA $FEE0,X     Adresse des Puffers holen (High-Byte)
C7D3    85 95      STA $95         und merken
C7D5    AD 88 FE   LDA $FE88       Position des Diskettennamens als
C7D8    85 94      STA $94         Low-Byte der Pufferadresse übernehmen
C7DA    A0 16      LDY #$16        Länge des Diskettennamens
C7DC    B1 94      LDA ($94),Y     ein Zeichen des Namens holen
```

```
C7DE  C9 A0       CMP #$A0         mit 'Shift Space' vergleichen
C7E0  D0 0B       BNE $C7ED        ist Diskettenname zu Ende ?
C7E2  A9 31       LDA #$31         ja, Dummy für nachfolgende Prüfung ('1')
C7E4  2C          .byte $2C        zwei Bytes überspringen
C7E5¹ B1 94       LDA ($94),Y      Zeichen aus Directoryeintrag holen
C7E7  C9 A0       CMP #$A0         mit 'Shift Space' vergleichen
C7E9  D0 02       BNE $C7ED        ist Eintrag zu Ende ?
C7EB  A9 20       LDA #$20         ja, Leerzeichen
C7ED² 99 B3 02    STA $02B3,Y      in Puffer übertragen und
C7F0  88          DEY              Pufferzeiger auf nächstes Byte setzen
C7F1  10 F2       BPL $C7E5        Ende des Puffers erreicht
C7F3  A9 12       LDA #$12         ja, Code für 'Revers On' vor an
C7F5  8D B1 02    STA $02B1        Zeilenbeginn in Puffer setzen
C7F8  A9 22       LDA #$22         Anführungszeichen vor
C7FA  8D B2 02    STA $02B2        und hinter den Diskettennamen
C7FD  8D C3 02    STA $02C3        setzen
C800  A9 20       LDA #$20         Leerzeichen in
C802  8D C4 02    STA $02C4        Puffer schreiben
C805  60          RTS              zurück zur aufrufenden Routine
-----------------------------------------------------------------------
[C6F2/C707]
Abschlußzeile mit 'Blocks free.' erstellen
C806  20 AC C7    JSR $C7AC        Puffer für Directoryzeile löschen
C809  A0 0B       LDY #$0B         Länge der Zeile setzen
C80B¹ B9 17 C8    LDA $C817,Y      Zeichen aus 'Blocks Free' String holen
C80E  99 B1 02    STA $02B1,Y      und in Puffer schreiben
C811  88          DEY              Pufferzeiger auf nächstes Byte setzen
C812  10 F7       BPL $C80B        Zeile fertig ?
C814  4C 4D EF    JMP $EF4D        ja, Zahl der freien Blöcke holen
-----------------------------------------------------------------------
C817  42 4C 4F 43 4B 53 20         'BLOCKS '
C81E  46 52 45 45 2E               'FREE.'
-----------------------------------------------------------------------
[Einsprung durch Routine C146]
Routine für Sratch-Befehl
C823  20 98 C3    JSR $C398        prüfen ob Befehl auf Dateityp beschränkt
C826  20 20 C3    JSR $C320        Laufwerksnummer aus Befehlsstring holen
C829  20 CA C3    JSR $C3CA        Laufwerk initialisieren
C82C  A9 00       LDA #$00         Zähler für Zahl der gelöschten
C82E  85 86       STA $86          Dateien zurücksetzen
C830  20 9D C4    JSR $C49D        ersten Dateieintrag holen
```

```
C833    30 3D     BMI $C872      Eintrag gefunden ?
C835[1] 20 B7 DD  JSR $DDB7      ja, Datei auf Gültigkeit testen
C838    90 33     BCC $C86D      wurde Datei richtig abgeschlossen ?
C83A    A0 00     LDY #$00       ja, Zeiger auf Position für Dateityp
C83C    B1 94     LDA ($94),Y    Dateityp aus Directory holen
C83E    29 40     AND #$40       Bit 6 als Flag für Scratchschutz testen
C840    D0 2B     BNE $C86D      ist die Datei löschgeschützt ?
C842    20 B6 C8  JSR $C8B6      nein, Eintrag löschen
C845    A0 13     LDY #$13       Zeiger auf Side-Sektoreintrag setzen
C847    B1 94     LDA ($94),Y    Spurnummer des ersten Side-Sektors holen
C849    F0 0A     BEQ $C855      Side-Sektor vorhanden ?
C84B    85 80     STA $80        ja, Spurnummer merken
C84D    C8        INY            und dazugehörige
C84E    B1 94     LDA ($94),Y    Sektornummer aus Eintrag holen
C850    85 81     STA $81        und merken
C852    20 7D C8  JSR $C87D      Blocks verfolgen und freigeben
C855[1] AE 53 02  LDX $0253      Nummer des Dateieintrags
C858    A9 20     LDA #$20       Flag für 'Eintrag nicht geschlossen'
C85A    35 E7     AND $E7,X      in Dateitypkennzeichen prüfen
C85C    D0 0D     BNE $C86B      ist Eintrag eine '*' Datei ?
C85E    BD 80 02  LDA $0280,X    nein, Spur des ersten Sektors
C861    85 80     STA $80        als aktuelle Spurnummer setzen
C863    BD 85 02  LDA $0285,X    Sektornummer der Dateidaten
C866    85 81     STA $81        übernehmen
C868    20 7D C8  JSR $C87D      Blocks der Datei verfolgen und freigeben
C86B[1] E6 86     INC $86        Zahl der gelöschten Dateien erhöhen
C86D[2] 20 8B C4  JSR $C48B      nächsten Dateieintrag holen
C870    10 C3     BPL $C835      gefunden ?
C872[1] A5 86     LDA $86        nein, Zahl der gelöschten Dateien
C874    85 80     STA $80        an Rückmeldung übergeben
C876    A9 01     LDA #$01       Nummer der Rückmeldung
C878    A0 00     LDY #$00       Wert für Sektornummer
C87A    4C A3 C1  JMP $C1A3      '01 Files Scratched' erzeugen
----------------------------------------------------------------------------
[C852/C868/DC1B]
Sektoren anhand der Verkettung verfolgen und in BAM freigeben
C87D    20 5F EF  JSR $EF5F      ersten, aktuellen Block freigeben
C880    20 75 D4  JSR $D475      nächsten Sektor lesen
C883    20 19 F1  JSR $F119      Nummer des BAM Kanals holen
C886    B5 A7     LDA $A7,X      Nummer des 2. Puffers holen
C888    C9 FF     CMP #$FF       mit 'Puffer frei' vergleichen
```

```
C88A   F0 08      BEQ $C894       ist Puffer zugewiesen ?
C88C   AD F9 02   LDA $02F9       nein, Flag in Zeiger für
C88F   09 40      ORA #$40        'BAM ungültig, auf Diskette schreiben'
C891   8D F9 02   STA $02F9       setzen
C894[2] A9 00     LDA #$00        Pufferzeiger
C896   20 C8 D4   JSR $D4C8       auf Beginn des Sektors setzen
C899   20 56 D1   JSR $D156       Byte aus Sektor holen
C89C   85 80      STA $80         und Spur des nächsten Sektors merken
C89E   20 56 D1   JSR $D156       Byte aus Sektor holen
C8A1   85 81      STA $81         und Nummer des nächsten Sektors setzen
C8A3   A5 80      LDA $80         Spurnummer des nächsten Sektors holen
C8A5   D0 06      BNE $C8AD       ist der Aktuelle der letzte Sektor ?
C8A7   20 F4 EE   JSR $EEF4       ja, BAM wieder auf Diskette schreiben
C8AA   4C 27 D2   JMP $D227       Kanal wieder schließen und Ende
C8AD[1] 20 5F EF  JSR $EF5F       Sektor in BAM freigeben
C8B0   20 4D D4   JSR $D44D       nächsten Sektor lesen
C8B3   4C 94 C8   JMP $C894       und weitermachen
-----------------------------------------------------------------------------
[C842/D8D3/EDDF]
Dateieintrag im Dateityp des Directory als gelöscht kennzeichnen
C8B6   A0 00      LDY #$00        Pufferzeiger auf Dateityp setzen
C8B8   98         TYA             Wert für Dateityp 'DEL'
C8B9   91 94      STA ($94),Y     in Eintrag übernehmen
C8BB   20 5E DE   JSR $DE5E       und Directory zurückschreiben
C8BE   4C 99 D5   JMP $D599       warten bis Schreiben ausgeführt
-----------------------------------------------------------------------------
[C909/Einsprung durch Routine C146]
Routine für Backup-Befehl (auf Einzellaufwerk nicht möglich)
C8C1   A9 31      LDA #$31        Fehlermeldung
C8C3   4C C8 C1   JMP $C1C8       '31 Syntax Error' ausgeben und zurück
-----------------------------------------------------------------------------
[A780]
Routine für 1541-New-Befehl (Diskette formatieren)
C8C6   A9 4C      LDA #$4C        In Pufferadresse $0600-$0602
C8C8   8D 00 06   STA $0600       wird ein JMP-Zeiger auf
C8CB   A9 C7      LDA #$C7        die Formatierungsroutine des
C8CD   8D 01 06   STA $0601       Diskcontrollers gesetzt ($FAC7), der vor
C8D0   A9 FA      LDA #$FA        jeder neuen Spur aufgerufen wird, womit
C8D2   8D 02 06   STA $0602       eigene Vorprogramme einbindbar sind
C8D5   A9 03      LDA #$03        Nummer des verwendeten Puffers
C8D7   20 D3 D6   JSR $D6D3       Spur- und Sektornummer an Jobschleife
```

```
C8DA  A5 7F      LDA $7F       Nummer des aktuellen Laufwerks holen
C8DC  09 E0      ORA #$E0      Jobcode für Pufferprogramm einbinden
C8DE  85 03      STA $03       an übergeben (Sprung zuf Zeiger)
C8E0  A5 03      LDA $03       und Rückmeldung holen
C8E2  30 FC      BMI $C8E0     warten bis Diskette formatiert ist
C8E4  C9 02      CMP #$02      Rückmeldung mit 'Ok' vergleichen
C8E6  90 07      BCC $C8EF     Formatieren fehlerlos beendet ?
C8E8  A9 03      LDA #$03      nein, Fehlernummer für 'File'
C8EA  A2 00      LDX #$00      Puffer 0 anwählen
C8EC  4C 0A E6   JMP $E60A     und Meldung ausgeben
C8EF  60         RTS           zurück zur aufrufenden Routine
------------------------------------------------------------------
[Einsprung durch Routine C146]
Routine für Copy-Befehl (Dateien kopieren)
C8F0  A9 E0      LDA #$E0      alle Puffer in
C8F2  8D 4F 02   STA $024F     Bitverzeichnis belegen
C8F5  20 D1 F0   JSR $F0D1     Spur- und Sektornummer für BAM setzen
C8F8  20 19 F1   JSR $F119     Puffernummer der BAM festlegen
C8FB  A9 FF      LDA #$FF      Puffer der BAM
C8FD  95 A7      STA $A7,X     als 'frei' Kennzeichen
C8FF  A9 0F      LDA #$0F      alle Kanäle in entsprechendem
C901  8D 56 02   STA $0256     Bitverzeichnis freigeben
C904  20 E5 C1   JSR $C1E5     ':' im Befehlsstring suchen
C907  D0 03      BNE $C90C     gefunden ?
C909  4C C1 C8   JMP $C8C1     nein, Fehlermeldung '31 Syntax Error'
C90C  20 F8 C1   JSR $C1F8     Eingabestring bearbeiten
C90F  20 20 C3   JSR $C320     Laufwerksnummer holen und setzen
C912  AD 8B 02   LDA $028B     Befehlssyntaxflag holen
C915  29 55      AND #$55      und Flags für Dateinamen holen
C917  D0 0F      BNE $C928     sind mehrere Dateinamen vorhanden ?
C919  AE 7A 02   LDX $027A     ja, Position des Zielnamens im Befehl
C91C  BD 00 02   LDA $0200,X   Zeichen aus Dateinamen holen
C91F  C9 2A      CMP #$2A      auf '*' Joker prüfen
C921  D0 05      BNE $C928     Joker vorhanden ?
C923  A9 30      LDA #$30      ja, Fehlermeldung
C925  4C C8 C1   JMP $C1C8     '30 Syntax Error' ausgeben
C928  AD 8B 02   LDA $028B     Befehlssyntaxflag holen
C92B  29 D9      AND #$D9      Flags für Joker testen
C92D  D0 F4      BNE $C923     sind Joker vorhanden ?
C92F  4C 52 C9   JMP $C952     nein, Datei kopieren
------------------------------------------------------------------
```

```
[Routine wird im DOS nicht verwendet]
Zeiger für Backup-Befehl initialisieren (Befehl nicht vorhanden)
C932  A9 00     LDA #$00      Zeiger löschen :
C934  8D 58 02  STA $0258     Länge eines Records
C937  8D 8C 02  STA $028C     Zahl der Zugrife auf Laufwerk
C93A  8D 80 02  STA $0280     Spurnummer für Zieldatei
C93D  8D 81 02  STA $0281     Spurnummer für Quelldatei
C940  A5 E3     LDA $E3       Wert für Standardlaufwerk
C942  29 01     AND #$01      Angabe auf ein Bit (0) beschränken
C944  85 7F     STA $7F       und in Zeiger für aktuelles Laufwerk
C946  09 01     ORA #$01      Nummer des aktuellen Directorysektors
C948  8D 91 02  STA $0291     zurücksetzen
C94B  AD 7B 02  LDA $027B     Position des zweiten Parameters an
C94E  8D 7A 02  STA $027A     erste Stelle kopieren
C951  60        RTS           zurück zur aufrufenden Routine
-----------------------------------------------------------------------
[C92F]
Datei(en) kopieren
C952  20 4F C4  JSR $C44F     Dateieintrag im Directory suchen
C955  AD 78 02  LDA $0278     Zahl der als Quelldatei genannten
C958  C9 03     CMP #$03      Einzeldateien
C95A  90 45     BCC $C9A1     weniger als 3 Dateien genannt ?
C95C  A5 E2     LDA $E2       ja, Laufwerksnummer der Zieldatei
C95E  C5 E3     CMP $E3       mit Laufwerk der Quelldatei vergleichen
C960  D0 3F     BNE $C9A1     nur auf einer Diskette kopieren ?
C962  A5 DD     LDA $DD       ja, Position der Zieldatei im Directory
C964  C5 DE     CMP $DE       mit Quelldatei vergleichen
C966  D0 39     BNE $C9A1     identisch ?
C968  A5 D8     LDA $D8       ja, Nummer des dazugehörigen Directory-
C96A  C5 D9     CMP $D9       sektors auf dem der Quelldatei prüfen
C96C  D0 33     BNE $C9A1     soll Eintrag überschrieben werden ?
C96E  20 CC CA  JSR $CACC     ja, Dateieintrag im Directory suchen
C971  A9 01     LDA #$01      Zeiger auf ersten
C973  8D 79 02  STA $0279     Dateinamen setzen
C976  20 FA C9  JSR $C9FA     Datei zum Lesen öffenen
C979  20 25 D1  JSR $D125     und Dateityp holen
C97C  F0 04     BEQ $C982     ist Dateieintrag eine Relative Datei ?
C97E  C9 02     CMP #$02      nein, auf Kennzeichen für 'PRG' testen
C980  D0 05     BNE $C987     identisch ?
C982  A9 64     LDA #$64      ja, Fehlermeldung
C984  20 C8 C1  JSR $C1C8     '64 File Type Mismatch' ausgeben
```

```
C987[1] A9 12      LDA #$12     Interner Schreibkanal (18)
C989  85 83        STA $83      setzen
C98B  AD 3C 02     LDA $023C    Nummer des zugewiesenen internen Kanals
C98E  8D 3D 02     STA $023D    in Lesekanal übertragen
C991  A9 FF        LDA #$FF     Wert für 'Kanal frei'
C993  8D 3C 02     STA $023C    in Tabelle setzen
C996  20 2A DA     JSR $DA2A    erste Quelldatei in Zieldatei kopieren
C999  A2 02        LDX #$02     Zeiger auf zweiten Dateinamen
C99B  20 B9 C9     JSR $C9B9    nächste Datei anhängen
C99E  4C 94 C1     JMP $C194    Befehl beenden und 'Ok' Meldung ausgeben
C9A1[4] 20 A7 C9   JSR $C9A7    Dateien kopieren
C9A4  4C 94 C1     JMP $C194    Befehl beenden und 'Ok' Meldung ausgeben
------------------------------------------------------------------------
[C9A1]
Einzeldatei kopieren
C9A7  20 E7 CA     JSR $CAE7    prüfen ob Eintrag schon existiert
C9AA  A5 E2        LDA $E2      Laufwerksnennung der Zieldatei holen
C9AC  29 01        AND #$01     und als Nummer des
C9AE  85 7F        STA $7F      aktuellen Laufwerks übernehmen
C9B0  20 86 D4     JSR $D486    internen Kanal zum Schreiben öffnen
C9B3  20 E4 D6     JSR $D6E4    Zieldatei im Directory eintragen
C9B6  AE 77 02     LDX $0277    Zahl der Zielnamen (1)
------------------------------------------------------------------------
[C99B/C9F1]
mehrere Dateien kopieren
C9B9  8E 79 02     STX $0279    als Quellnamenzahl übernehmen
C9BC  20 FA C9     JSR $C9FA    Directory lesen
C9BF  A9 11        LDA #$11     16 (Nummer des internen Lesekanals)
C9C1  85 83        STA $83      als aktuelle Sekundäradresse festlegen
C9C3  20 EB D0     JSR $D0EB    Kanal öffnen
C9C6  20 25 D1     JSR $D125    Dateityp des Eintrags holen
C9C9  D0 03        BNE $C9CE    ist Datei eine Relative Datei ?
C9CB  20 53 CA     JSR $CA53    ja, Relative Datei kopieren
C9CE[1] A9 08      LDA #$08     Flag für letztes Zeichen (EOI)
C9D0  85 F8        STA $F8      setzen
C9D2  4C D8 C9     JMP $C9D8    und Kopierarbeit abschließen
C9D5[1] 20 9B CF   JSR $CF9B    Byte in Zieldatei schreiben
C9D8[1] 20 35 CA   JSR $CA35    Byte aus Quelldatei holen
C9DB  A9 80        LDA #$80     Flag für EOI (letztes Zeichen)
C9DD  20 A6 DD     JSR $DDA6    testen
C9E0  F0 F3        BEQ $C9D5    ist Flag gesetzt ?
```

```
C9E2  20 25 D1   JSR $D125      ja, Dateityp holen
C9E5  F0 03      BEQ $C9EA      ist Dateieintrag eine Relative Datei ?
C9E7  20 9B CF   JSR $CF9B      nein, Byte in Datei schreiben
C9EA  AE 79 02   LDX $0279      Nummer der Quelldatei
C9ED  E8         INX            mit Zahl der
C9EE  EC 78 02   CPX $0278      Quelldateien vergleichen
C9F1  90 C6      BCC $C9B9      sind weitere Dateien angegeben
C9F3  A9 12      LDA #$12       Nummer für Schreibkanal (18)
C9F5  85 83      STA $83        als aktuelle Sekundäradresse setzen
C9F7  4C 02 DB   JMP $DB02      Datei und Kanal schließen
-----------------------------------------------------------------

[C976/C9BC]
Kanal um Datei zu lesen öffnen
C9FA  AE 79 02   LDX $0279      Nummer des Dateinamens holen
C9FD  B5 E2      LDA $E2,X      entsprechende Laufwerksnummer
C9FF  29 01      AND #$01       herstellen und
CA01  85 7F      STA $7F        als aktuelles Laufwerk merken
CA03  AD 85 FE   LDA $FE85      Nummer der Directoryspur (18)
CA06  85 80      STA $80        als aktuelle Spur festlegen
CA08  B5 D8      LDA $D8,X      Directorysektor des Eintrags feststellen
CA0A  85 81      STA $81        als aktuellen Sektor setzen
CA0C  20 75 D4   JSR $D475      Sektor in Puffer einlesen
CA0F  AE 79 02   LDX $0279      Nummer der Dateinennung im Befehl
CA12  B5 DD      LDA $DD,X      dazugehörigen Zeiger auf Directory-
CA14  20 C8 D4   JSR $D4C8      position holen und Pufferzeiger setzen
CA17  AE 79 02   LDX $0279      Nummer der Dateinennungen des Befehls
CA1A  B5 E7      LDA $E7,X      entsprechendes Dateitypkennzeichen
CA1C  29 07      AND #$07       holen und Dateityp daraus herstellen
CA1E  8D 4A 02   STA $024A      und merken
CA21  A9 00      LDA #$00       Zeiger für Recordlänge der
CA23  8D 58 02   STA $0258      Datei löschen
CA26  20 A0 D9   JSR $D9A0      Datei zum lesen öffnen
CA29  A0 01      LDY #$01       Pufferzeiger setzen
CA2B  20 25 D1   JSR $D125      Dateityp holen
CA2E  F0 01      BEQ $CA31      ist Datei eine Relative Datei ?
CA30  C8         INY            nein, Pufferzeiger auf nächstes Byte
CA31  98         TYA            (Spurnummer)
CA32  4C C8 D4   JMP $D4C8      Pufferzeiger initialisieren
-----------------------------------------------------------------
```

```
[C9D8/E81B/E839]
Ein Byte aus Datei lesen
CA35  A9 11     LDA #$11     interne Kanalnummer
CA37  85 83     STA $83      für Lesen setzen
CA39  20 9B D3  JSR $D39B    ein Byte lesen
CA3C  85 85     STA $85      und merken
CA3E  A6 82     LDX $82      Nummer des Kanals holen und
CA40  B5 F2     LDA $F2,X    dazugehörigen Kanalstatus feststellen
CA42  29 08     AND #$08     Bitflag für 'letztes Byte' (EOI)
CA44  85 F8     STA $F8      herstellen und merken
CA46  D0 0A     BNE $CA52    ist die Datei zu Ende ?
CA48  20 25 D1  JSR $D125    nein, Dateityp holen
CA4B  F0 05     BEQ $CA52    ist es eine Relative Datei ?
CA4D  A9 80     LDA #$80     nein,
CA4F  20 97 DD  JSR $DD97    alle entsprechenden Flags setzen
CA52  60        RTS          zurück zur aufrufenden Routine
-------------------------------------------------------------------
[C9CB]
Realtive Datei kopieren
CA53  20 D3 D1  JSR $D1D3    aktuelle Laufwerksnummer setzen
CA56  20 CB E1  JSR $E1CB    Position des letzten Records holen
CA59  A5 D6     LDA $D6      Position in Side-Sektor
CA5B  48        PHA          merken
CA5C  A5 D5     LDA $D5      und Nummer des entsprechenden
CA5E  48        PHA          Side-Sektors festhalten
CA5F  A9 12     LDA #$12     interner Kanal für Schreiben
CA61  85 83     STA $83      setzen
CA63  20 07 D1  JSR $D107    Kanal öffnen
CA66  20 D3 D1  JSR $D1D3    aktuelle Laufwerksnummer setzen
CA69  20 CB E1  JSR $E1CB    Position des letzten Side-Sektors holen
CA6C  20 9C E2  JSR $E29C    und Sektor in Puffer lesen
CA6F  A5 D6     LDA $D6      aktuellen Zeiger auf Position
CA71  85 87     STA $87      in Side-Sektor merken
CA73  A5 D5     LDA $D5      Nummer des Side-Sektors
CA75  85 86     STA $86      merken
CA77  A9 00     LDA #$00     Zeiger löschen :
CA79  85 88     STA $88      Zwischenspeicher
CA7B  85 D4     STA $D4      Zeiger auf Beginn des Records
CA7D  85 D7     STA $D7      Zeiger auf Position in Record
CA7F  68        PLA          Nummer des lezten Side-Sektors
CA80  85 D5     STA $D5      holen und setzen
```

```
CA82  68        PLA            Nummer des letzten Recordeintrags
CA83  85 D6     STA $D6        im Side-Sektor holen und setzen
CA85  4C 3B E3  JMP $E33B      Side-Sektoren aktualisieren
----------------------------------------------------------------------
[Einsprung über Routine C146]
Routine für Rename-Befehl
CA88  20 20 C3  JSR $C320      Nummer des Laufwerks holen
CA8B  A5 E3     LDA $E3        Nummer des Standardlaufwerks
CA8D  29 01     AND #$01       herstellen und wieder
CA8F  85 E3     STA $E3        setzen
CA91  C5 E2     CMP $E2        mit letzter Laufwerksnummer vergleichen
CA93  F0 02     BEQ $CA97      muss Laufwerk gewechselt werden ?
CA95  09 80     ORA #$80       ja, Bitflag für Suche auf beiden
CA97  85 E2     STA $E2        Laufwerken setzen
CA99  20 4F C4  JSR $C44F      neuer Name in Directory suchen
CA9C  20 E7 CA  JSR $CAE7      Name schon vorhanden ?
CA9F  A5 E3     LDA $E3        Nummer des Standardlaufwerks
CAA1  29 01     AND #$01       herstellen und als Nummer
CAA3  85 7F     STA $7F        des aktuellen Laufwerks übernehmen
CAA5  A5 D9     LDA $D9        Nummer des Directorysektors
CAA7  85 81     STA $81        setzen
CAA9  20 57 DE  JSR $DE57      und Sektor in Puffer einlesen
CAAC  20 99 D5  JSR $D599      und warten bis ausgeführt
CAAF  A5 DE     LDA $DE        Zeiger auf Directoryeintrag auf
CAB1  18        CLC            Startposition des
CAB2  69 03     ADC #$03       Dateinamens im Directory setzen
CAB4  20 C8 D4  JSR $D4C8      Pufferzeiger festlegen
CAB7  20 93 DF  JSR $DF93      Nummer des aktuellen Puffers holen
CABA  A8        TAY            und merken
CABB  AE 7A 02  LDX $027A      Position des neuen Namens im Befehl
CABE  A9 10     LDA #$10       max. Länge des Dateinamens
CAC0  20 6E C6  JSR $C66E      Namen in aus Befehlsstring in Puffer
CAC3  20 5E DE  JSR $DE5E      Directorysektor wieder schreiben
CAC6  20 99 D5  JSR $D599      und warten bis ausgeführt
CAC9  4C 94 C1  JMP $C194      Rückmeldung bereitstellen und beenden
----------------------------------------------------------------------
[C96E/CAE7]
Prüft ob Dateieintrag vorhanden ist
CACC  A5 E8     LDA $E8        Dateityp des 2. Namens holen
CACE  29 07     AND #$07       und Kennzeichen für Typ isolieren
CAD0  8D 4A 02  STA $024A      als aktuellen Dateityp merken
```

```
CAD3  AE 78 02   LDX $0278       Startposition des Dateinamens im
CAD6  CA         DEX             Befehlsstring holen
CAD7  EC 77 02   CPX $0277       mit Start des Befehlsstrings vergleichen
CADA  90 0A      BCC $CAE6       sind noch Zeichen vor Dateinamen ?
CADC  BD 80 02   LDA $0280,X     ja, Sektornummer der Datei holen
CADF  D0 F5      BNE $CAD6       war das der letzte Sektor ?
CAE1  A9 62      LDA #$62        ja, Fehlermeldung
CAE3  4C C8 C1   JMP $C1C8       '62 File Not Found' ausgeben
CAE6  60         RTS             zurück zur aufrufenden Routine
```

[C9A7/CA9C]

Zwei Dateinamen vergleichen

```
CAE7  20 CC CA   JSR $CACC       Datei im Directory vorhanden ?
CAEA  BD 80 02   LDA $0280,X     Nummer des ersten Datensektors holen
CAED  F0 05      BEQ $CAF4       ist Sektor vorhanden ?
CAEF  A9 63      LDA #$63        ja, Fehlermeldung
CAF1  4C C8 C1   JMP $C1C8       '63 File exist' ausgeben
CAF4  CA         DEX             nächsten Namen anwählen
CAF5  10 F3      BPL $CAEA       war das der letzte Dateiname ?
CAF7  60         RTS             ja, zurück zur aufrufenden Routine
```

[Einsprung über Routine C146]

Routine für Memory-Befehl

```
CAF8  AD 01 02   LDA $0201       zweites Zeichen des Befehls holen
CAFB  C9 2D      CMP #$2D        mit '-' vergleichen
CAFD  D0 4C      BNE $CB4B       identisch ?
CAFF  AD 03 02   LDA $0203       ja, dann viertes Zeichen holen und als
CB02  85 6F      STA $6F         Speicheradresse (Low-Byte) merken
CB04  AD 04 02   LDA $0204       fünftes Zeichen holen und als
CB07  85 70      STA $70         Speicheradresse (High-Byte) merken
CB09  A0 00      LDY #$00        Pufferzeiger löschen
CB0B  AD 02 02   LDA $0202       drittes Zeichen des Befehls holen
CB0E  C9 52      CMP #$52        und mit 'R' vergleichen
CB10  F0 0E      BEQ $CB20       soll Read-Befehl durchgeführt werden ?
CB12  20 58 F2   JSR $F258       nein, Aufruf hat keine Funktion (RTS)
CB15  C9 57      CMP #$57        mit 'W' vergleichen
CB17  F0 37      BEQ $CB50       soll Write-Befehl durchgeführt werden ?
CB19  C9 45      CMP #$45        nein, mit 'E' vergleichen
CB1B  D0 2E      BNE $CB4B       soll Programm ausgeführt werden ?
CB1D  6C 6F 00   JMP ($006F)     ja, Programm starten
```

```
[CB10]
Memory-Read-Befehl ('M-R'); Liest Byte aus Speicher
CB20  B1 6F     LDA ($6F),Y   Byte aus angegebener Adresse holen
CB22  85 85     STA $85       und merken
CB24  AD 74 02  LDA $0274     Länge des Befehlsstrings holen
CB27  C9 06     CMP #$06      und mit maximaler Länge vergleichen
CB29  90 1A     BCC $CB45     ist der String kleiner ?
CB2B  AE 05 02  LDX $0205     nein, Zahl der zu lesenden Bytes holen
CB2E  CA        DEX           und korregieren (da eins schon gelesen)
CB2F  F0 14     BEQ $CB45     noch weitere Bytes aus Speicher lesen ?
CB31  8A        TXA           ja, Zeiger
CB32  18        CLC           mit Startadresse verrechnen
CB33  65 6F     ADC $6F       und so Endadresse des Bereichs errechnen
CB35  E6 6F     INC $6F       Zeiger auf aktuelles Byte erhöhen
CB37  8D 49 02  STA $0249     Endadresse (Low-Byte) merken
CB3A  A5 6F     LDA $6F       Zeiger auf aktuelle
CB3C  85 A5     STA $A5       Speicher als Zeiger auf Fehlermeldungs-
CB3E  A5 70     LDA $70       puffer für nachfolgende Routine ($D43A)
CB40  85 A6     STA $A6       übernehmen
CB42  4C 43 D4  JMP $D443     erstes Byte und Flags für Ausgabe setzen
CB45  20 EB D0  JSR $D0EB     Kanal suchen und öffnen
CB48  4C 3A D4  JMP $D43A     weitere Bytes ausgeben
CB4B  A9 31     LDA #$31      Fehlermeldung
CB4D  4C C8 C1  JMP $C1C8     '31 Syntax Error' ausgeben
--------------------------------------------------------------------------
[CB17/CB59]
Memory-Write-Befehl ('M-W'); Schreibt in Speicher
CB50  B9 06 02  LDA $0206,Y   Bytewert aus Befehlsstring holen
CB53  91 6F     STA ($6F),Y   und in Speicher schreiben
CB55  C8        INY           Pufferzeiger auf nächstes Byte richten
CB56  CC 05 02  CPY $0205     mit Wert für Ende vergleichen
CB59  90 F5     BCC $CB50     noch weitere Bytes übernehmen ?
CB5B  60        RTS           nein, zurück zur aufrufenden Routine
--------------------------------------------------------------------------
[Einsprung über Routine C146]
User-Befehl ('UX'); Startet Programm im DOS-Puffer
CB5C  AC 01 02  LDY $0201     zweites Zeichen des Befehls holen
CB5F  C0 30     CPY #$30      und mit '0' vergleichen
CB61  D0 09     BNE $CB6C     identisch ?
--------------------------------------------------------------------------
```

```
[EBBC]
User-Befehle ausführen
CB63    4C 26 80   JMP $8030       ja, User0-Befehle abfragen
CB66    EA         NOP             unbenutzter Leerbereich
CB67    EA         NOP             der durch Modifizierung
CB68    EA         NOP             der User-Routine des ROM
CB69    EA         NOP             1541-Laufwerks zur
CB6A    EA         NOP             1571-User-Routine
CB6B    EA         NOP             entstand
CB6C[1] 20 72 CB   JSR $CB72       Adresse setzen und Programm ausführen
CB6F    4C 94 C1   JMP $C194       bei Rücksprung mit 'RTS' Befehl beenden
CB72[1] 88         DEY             ASCII-Nummer des Befehls
CB73    98         TYA             in Binärnummer
CB74    29 0F      AND #$0F        umrechnen,
CB76    0A         ASL A           verdoppeln (Adresse ist 2-Byte-Zeiger)
CB77    A8         TAY             und merken
CB78    B1 6B      LDA ($6B),Y     zum Befehl gehörige Adresse (Low-Byte)
CB7A    85 75      STA $75         holen und merken
CB7C    C8         INY             Zeiger auf nächstes Byte der Adresse
CB7D    B1 6B      LDA ($6B),Y     High-Byte der Startadresse holen
CB7F    85 76      STA $76         und merken
CB81    4C 2D AA   JMP $AA2D       Programm starten
-----------------------------------------------------------------
[D819]
'#'-Befehl; Öffnet Direktzugriffskanal
CB84    AD 8E 02   LDA $028E       Laufwerksnummer des letzten Jobs
CB87    85 7F      STA $7F         als aktuelles Laufwerk setzen
CB89    A5 83      LDA $83         Kanalnummer holen
CB8B    48         PHA             und merken
CB8C    20 3D C6   JSR $C63D       Laufwerk initialisieren
CB8F    68         PLA             Kanalnummer wieder
CB90    85 83      STA $83         zurücksetzen
CB92    AE 74 02   LDX $0274       Länge des Befehlsstrings
CB95    CA         DEX             mit 1 vergleichen
CB96    D0 0D      BNE $CBA5       ist ein gewünschter Puffer angegeben ?
CB98    A9 01      LDA #$01        nein, Zahl der benötigten Puffer
CB9A    20 E2 D1   JSR $D1E2       freien Puffer und Kanal belegen
CB9D    4C F1 CB   JMP $CBF1       Zeiger und Tabellen initialisieren
CBA0[3] A9 70      LDA #$70        Fehlermeldung
CBA2    4C C8 C1   JMP $C1C8       '70 No Channel' ausgeben
CBA5[1] A0 01      LDY #$01        Zeiger auf Position in Puffer
```

```
CBA7    20 7C CC   JSR $CC7C     Byte aus Befehlsstring holen
CBAA    AE 85 02   LDX $0285     Puffernummer holen
CBAD    E0 05      CPX #$05      und mit maximalem Puffer vergleichen
CBAF    B0 EF      BCS $CBA0     ist die angegebene Nummer erlaubt (<5) ?
CBB1    A9 00      LDA #$00      ja, Zwischenspeicher
CBB3    85 6F      STA $6F       in Zeropage
CBB5    85 70      STA $70       löschen
CBB7    38         SEC           Flag für 'Puffer belegt'
CBB8[1] 26 6F      ROL $6F       in Zwischenspeicher
CBBA    26 70      ROL $70       einschieben
CBBC    CA         DEX           Nummer des Puffers
CBBD    10 F9      BPL $CBB8     ist Flag an richtiger Posiition ?
CBBF    A5 6F      LDA $6F       ja, errechnete Pufferbelegung
CBC1    2D 4F 02   AND $024F     mit Bitverzeichnis vergleichen
CBC4    D0 DA      BNE $CBA0     ist Puffer schon belegt ?
CBC6    A5 70      LDA $70       nein, Puffernummer 8-15
CBC8    2D 50 02   AND $0250     (nur bei CBM3030-CBM8250) testen
CBCB    D0 D3      BNE $CBA0     ist Puffer frei ?
CBCD    A5 6F      LDA $6F       ja, Bit für Puffer
CBCF    0D 4F 02   ORA $024F     in Bitverzeichnis übernehmen und
CBD2    8D 4F 02   STA $024F     dadurch Puffer belegen
CBD5    A5 70      LDA $70       dasselbe für Puffer 8-15
CBD7    0D 50 02   ORA $0250     (kommt nie vor, da nur 5 Puffer
CBDA    8D 50 02   STA $0250     vohanden sind)
CBDD    A9 00      LDA #$00      Zahl der Puffer auf 1 setzen
CBDF    20 E2 D1   JSR $D1E2     und Puffer und Kanal belegen
CBE2    A6 82      LDX $82       aktuelle Kanalnummer
CBE4    AD 85 02   LDA $0285     aktuelle Sektornummer
CBE7    95 A7      STA $A7,X     in Sektortabelle dem Kanal zuordnen
CBE9    AA         TAX           Zeiger korregieren
CBEA    A5 7F      LDA $7F       aktuelle Laufwerksnummer
CBEC    95 00      STA $00,X     als Jobcode
CBEE    9D 5B 02   STA $025B,X   übergeben
CBF1[1] A6 83      LDX $83       Nummer der Sekundäradresse feststellen
CBF3    BD 2B 02   LDA $022B,X   und zugeordneten interen Kanal holen
CBF6    09 40      ORA #$40      Kanal als 'aktiv'
CBF8    9D 2B 02   STA $022B,X   kennzeichnen
CBFB    A4 82      LDY $82       Nummer des aktuellen Kanals
CBFD    A9 FF      LDA #$FF      Zahl der zu übertragenden Daten
CBFF    99 44 02   STA $0244,Y   dem Kanal zuordnen
CC02    A9 89      LDA #$89      Kanal für Schreiben und Lesen
```

```
CC04  99 F2 00   STA $00F2,Y     freigeben
CC07  B9 A7 00   LDA $00A7,Y     Nummer des Puffers holen
CC0A  99 3E 02   STA $023E,Y     als auszugebendes Zeichen setzen
CC0D  0A         ASL A           Nummer verdoppeln
CC0E  AA         TAX             (da Tabelle 2-Byte-Werte hat)
CC0F  A9 01      LDA #$01        Pufferzeiger auf Beginn des Puffers
CC11  95 99      STA $99,X       setzen
CC13  A9 0E      LDA #$0E        Kennzeichen für Direktzugriff
CC15  99 EC 00   STA $00EC,Y     in Dateityptabelle verzeichnen
CC18  4C 94 C1   JMP $C194       Rückmeldung ausgeben und beenden
------------------------------------------------------------------------
[Einsprung über Routine C146]
Routine für Block-Befehle
CC1B  A0 00      LDY #$00        Startposition im Eingabepuffer setzen
CC1D  A2 00      LDX #$00        Zeiger auf Zahl der Parameter löschen
CC1F  A9 2D      LDA #$2D        '-' als zu suchendes Zeichen setzen
CC21  20 68 C2   JSR $C268       Eingabestring bearbeiten
CC24  D0 0A      BNE $CC30       Zeichen gefunden ?
CC26  A9 31      LDA #$31        nein, Fehlermeldung
CC28  4C C8 C1   JMP $C1C8       '31 Syntax Error' ausgeben
CC2B  A9 30      LDA #$30        Fehlermeldung
CC2D  4C C8 C1   JMP $C1C8       '30 Syntax Error' ausgeben
CC30  8A         TXA             Zahl der gefundenen Parameter
CC31  D0 F8      BNE $CC2B       wurden weitere Angaben gefunden ?
CC33  A2 05      LDX #$05        ja, Zeiger in Eingabepuffer setzen
CC35  B9 00 02   LDA $0200,Y     drittes Zeichen aus Puffer holen
CC38  DD 5D CC   CMP $CC5D,X     und mit Blockbefehl vergleichen
CC3B  F0 05      BEQ $CC42       liegt ein Blockbefehl vor ?
CC3D  CA         DEX             nein, Zeiger auf nächsten Befehl setzen
CC3E  10 F8      BPL $CC38       bereits mit allen Befehlen verglichen ?
CC40  30 E4      BMI $CC26       ja, immer Sprung nach $CC26
CC42  8A         TXA             Nummer des Blockbefehls
CC43  09 80      ORA #$80        Flag für 'erweiterter Befehl'
CC45  8D 2A 02   STA $022A       merken
CC48  20 6F CC   JSR $CC6F       Parameter des Befehls holen und prüfen
CC4B  AD 2A 02   LDA $022A       Befehlsnummer wieder holen
CC4E  0A         ASL A           und verdoppeln
CC4F  AA         TAX             (da in Adresstabelle aus 2-Byte Zeiger)
CC50  BD 64 CC   LDA $CC64,X     Startadresse des Befehls (Low-Byte)
CC53  85 70      STA $70         holen und merken
CC55  BD 63 CC   LDA $CC63,X     High-Byte holen und in Zeiger
```

```
CC58  85 6F      STA $6F         übernehmen
CC5A  6C 6F 00   JMP ($006F)     Blockbefehl starten
-----------------------------------------------------------------------
[CC38]  Befehlkodes der Block-Befehle
CC5D  41 46 52 57 45 50          'A' , 'F' , 'R' , 'W' , 'E' , 'P'
-----------------------------------------------------------------------
[CC50/CC55]  Startadressen der Blockbefehlsroutinen
CC63  03 CD                      $CD03  B-A Befehl
CC65  F5 CC                      $CCF5  B-F Befehl
CC67  56 CD                      $CD56  B-R Befehl
CC69  73 CD                      $CD73  B-W Befehl
CC6B  A3 CD                      $CDA3  B-E Befehl
CC6D  BD CD                      $CDBD  B-P Befehl
-----------------------------------------------------------------------
[CC48/CD5F/CD97]
Parameter für Blockbefehle holen und setzen
CC6F  A0 00      LDY #$00        Startposition der Suche im Befehlsstring
CC71  A2 00      LDX #$00        Zahl der gefundenen Parameter löschen
CC73  A9 3A      LDA #$3A        ':' als zu suchendes Zeichen setzen
CC75  20 68 C2   JSR $C268       und im Eingabepuffer suchen
CC78  D0 02      BNE $CC7C       Zeichen gefunden ?
CC7A  A0 03      LDY #$03        nein, Pufferzeiger auf viertes Zeichen
-----------------------------------------------------------------------
[CBA7/CC78/CC8F]
Parameter für Blockbefehl testen
CC7C  B9 00 02   LDA $0200,Y     und Zeichen holen
CC7F  C9 20      CMP #$20        mit Wert für Leerzeichen ' ' vergleichen
CC81  F0 08      BEQ $CC8B       identisch ?
CC83  C9 1D      CMP #$1D        nein, auf Wert für 'Cursor right' prüfen
CC85  F0 04      BEQ $CC8B       identisch ?
CC87  C9 2C      CMP #$2C        nein, mit Wert für Komma vergleichen
CC89  D0 07      BNE $CC92       identisch ?
CC8B  C8         INY             ja, Pufferzeiger auf nächstes Zeichen
CC8C  CC 74 02   CPY $0274       auf Wert für Befehlsstringende prüfen
CC8F  90 EB      BCC $CC7C       ist Zeiger am Ende des Eingabepuffers ?
CC91  60         RTS             ja, zurück zur aufrufenden Routine
CC92  20 A1 CC   JSR $CCA1       Parameter holen, umrechnen und setzen
CC95  EE 77 02   INC $0277       aktuelle Zahl der Parameter
CC98  AC 79 02   LDY $0279       Gesamtzahl der Parameter
CC9B  E0 04      CPX #$04        mit maximaler Zahl der Parameter prüfen
CC9D  90 EC      BCC $CC8B       zu viele Parameterangaben ?
```

```
CC9F   B0 8A      BCS $CC2B        ja, immer Sprung nach $CC2B
--------------------------------------------------------------------
```

[CC92]

Parameter der Blockbefehle von ASCII- in Binärwerte umrechnen und setzen

```
CCA1    A9 00      LDA #$00         Bereich, der als
CCA3    85 6F      STA $6F          Zwischenspeicher für
CCA5    85 70      STA $70          Rechenoperationen verwendet wird,
CCA7    85 72      STA $72          löschen
CCA9    A2 FF      LDX #$FF         Zeiger auf aktuelles Rechenregister
CCAB[1] B9 00 02   LDA $0200,Y      nächstes Zeichen aus Eingabepuffer holen
CCAE    C9 40      CMP #$40         mit ASCII-Wert von '@' vergleichen
CCB0    B0 18      BCS $CCCA        liegt ein Buchstabe vor ?
CCB2    C9 30      CMP #$30         nein, auf Wert für '0' prüfen
CCB4    90 14      BCC $CCCA        liegt eine Ziffer vor ?
CCB6    29 0F      AND #$0F         binären Zahlenwert berechnen
CCB8    48         PHA              und merken
CCB9    A5 70      LDA $70          Werte im Zwischenspeicher
CCBB    85 71      STA $71          von $6F bis $71 um eine
CCBD    A5 6F      LDA $6F          Stelle in Richtung $71 verschieben,
CCBF    85 70      STA $70          sodaß $6F frei wird
CCC1    68         PLA              binären Zahlenwert wieder holen und
CCC2    85 6F      STA $6F          in Zwischenspeicher schreiben
CCC4    C8         INY              Pufferzeiger auf nächstes Zeichen
CCC5    CC 74 02   CPY $0274        auf Endposition des Parameters prüfen
CCC8    90 E1      BCC $CCAB        gesamte Dezimalzahl eingelesen ?
CCCA[2] 8C 79 02   STY $0279        ja, aktueller Pufferzeiger merken
CCCD    18         CLC              Addierroutine initialisieren
CCCE    A9 00      LDA #$00         'Dummy-Wert' für ersten Durchlauf der
CCD0[1] E8         INX              Routine
CCD1    E0 03      CPX #$03         auf max. Zahl der Dezimalstellen prüfen
CCD3    B0 0F      BCS $CCE4        sind zu viele Ziffern angegeben ?
CCD5    B4 6F      LDY $6F,X        nein, Wert einer Ziffer in Zähler holen
CCD7[2] 88         DEY              Zähler erniedrigen
CCD8    30 F6      BMI $CCD0        ist Dezimalziffer Null ?
CCDA    7D F2 CC   ADC $CCF2,X      nein, Binärwert des Ziffernwerts holen
CCDD    90 F8      BCC $CCD7        addieren; ist Binärzahl schon > 256 ?
CCDF    18         CLC              ja, Highbyte einrichten
CCE0    E6 72      INC $72          und um eins erhöhen
CCE2    D0 F3      BNE $CCD7        immer Sprung nach $CCD7
CCE4[1] 48         PHA              äquivalenten Binärwert (Low-Byte) merken
CCE5    AE 77 02   LDX $0277        Nummer des Parameters holen
```

```
CCE8  A5 72      LDA $72        errechneten Binärwert (High-Byte)
CCEA  9D 80 02   STA $0280,X    in Tabelle der Parameter eintragen
CCED  68         PLA            Low-Byte des Binärwerts wieder holen
CCEE  9D 85 02   STA $0285,X    und ebenfalls merken
CCF1  60         RTS            zurück zur aufrufenden Routine
-----------------------------------------------------------------
CCF2  01 0A 64                  Binärwerte für 1, 10 und 100
-----------------------------------------------------------------
[Einsprung über Routine CC1B]
Block-Free-Befehl ('B-F'); Block in BAM freigeben
CCF5  20 F5 CD   JSR $CDF5      Spur- und Sektornummer holen
CCF8  20 5F EF   JSR $EF5F      Bit für Block auf 'frei' setzen
CCFB  4C 94 C1   JMP $C194      Rückmeldung bereitstellen und beenden
-----------------------------------------------------------------
CCFE  A9 01      LDA #$01       unbenutzter Programmrest aus
CD00  8D F9 02   STA $02F9      CBM 4040 ROM
-----------------------------------------------------------------
[Einsprung über Routine CC1B]
Block-Allocate-Befehl ('B-A'); Block in BAM belegen
CD03  20 F5 CD   JSR $CDF5      Spur- und Sektornummer holen
CD06  A5 81      LDA $81        Sektornummer holen
CD08  48         PHA            und merken
CD09  20 FA F1   JSR $F1FA      nächsten freien Sektor in BAM suchen
CD0C  F0 0B      BEQ $CD19      ist Block frei gewesen ?
CD0E  68         PLA            ja, Nummer des gewünschten Sektors holen
CD0F  C5 81      CMP $81        mit akktueller Sektornummer vergleichen
CD11  D0 19      BNE $CD2C      identisch ?
CD13  20 90 EF   JSR $EF90      Sektor in BAM als belegt kennzeichnen
CD16  4C 94 C1   JMP $C194      Rückmeldung bereitstellen und beenden
CD19[1] 68       PLA            Stack korregieren, Sektornummer löschen
CD1A[1] A9 00    LDA #$00       Nummer des Sektors neu
CD1C  85 81      STA $81        festlegen
CD1E  E6 80      INC $80        Zeiger auf Spur auf nächste Spur setzen
CD20  A5 80      LDA $80        und holen
CD22  CD AC 02   CMP $02AC      mit Wert der größten Spur +1 vergleichen
CD25  B0 0A      BCS $CD31      ist Spurnummer kleiner ?
CD27  20 FA F1   JSR $F1FA      ja, nächsten freien Sektor suchen
CD2A  F0 EE      BEQ $CD1A      gefunden ?
CD2C[1] A9 65    LDA #$65       nein, Fehlermeldung
CD2E  20 45 E6   JSR $E645      '65 No Block' ausgeben
CD31[1] A9 65    LDA #$65       Fehlermeldung
```

```
CD33  20 C8 C1   JSR $C1C8        '65 No Block' bereitstellen
------------------------------------------------------------------------
[CD42/CDA6]
Parameter des 'B-R' Befehls prüfen und Sektor in Puffer lesen
CD36  20 F2 CD   JSR $CDF2        Spur- und Sektornummer prüfen und holen
CD39  4C 60 D4   JMP $D460        Sektor in Puffer lesen
------------------------------------------------------------------------
[CD4A]
Byte aus Puffer holen
CD3C  20 2F D1   JSR $D12F        Pufferzeiger setzen
CD3F  A1 99      LDA ($99,X)      Byte holen
CD41  60         RTS              zurück zur aufrufenden Routine
------------------------------------------------------------------------
[CD56/CD62]
Sektor von Diskette in Puffer lesen und Zeiger initialisieren
CD42  20 36 CD   JSR $CD36        Parameter holen und Sektor lesen
CD45  A9 00      LDA #$00         Position des Pufferzeigers festlegen
CD47  20 C8 D4   JSR $D4C8        Pufferzeiger setzen
CD4A  20 3C CD   JSR $CD3C        ein Byte aus Puffer holen
CD4D  99 44 02   STA $0244,Y      Zahl der noch zu übertragenden Daten
CD50  A9 89      LDA #$89         Kanal für Lesen und Schreiben
CD52  99 F2 00   STA $00F2,Y      freigeben
CD55  60         RTS              zurück zur aufrufenden Routine
------------------------------------------------------------------------
[Einsprung über Routine CC1B]
Routine für Block-Read-Befehl ('B-R'); Sektor von Diskette lesen
CD56  20 42 CD   JSR $CD42        Sektor lesen und Zeiger setzen
CD59  20 EC D3   JSR $D3EC        Byte aus Puffer ausgeben
CD5C  4C 94 C1   JMP $C194        Rückmeldung bereitstellen und beenden
------------------------------------------------------------------------
[Vektor: FFEA]
Routine für U1-Befehl (vgl. B-R); Sektor von Diskette lesen
CD5F  20 6F CC   JSR $CC6F        Parameterangaben holen
CD62  20 42 CD   JSR $CD42        Sektor in Puffer lesen
CD65  B9 44 02   LDA $0244,Y      Zahl der noch zu übertragenden Bytes
CD68  99 3E 02   STA $023E,Y      als auszugebendes Byte setzen
CD6B  A9 FF      LDA #$FF         Zahl der noch zu übertragenden Bytes
CD6D  99 44 02   STA $0244,Y      neu initialisieren
CD70  4C 94 C1   JMP $C194        Rückmeldung bereitstellen und beenden
------------------------------------------------------------------------
```

```
[Einsprung über Routine CC1B]
Routine für Block-Write-Befehl
CD73   20 F2 CD   JSR $CDF2        Puffer belegen und Kanal öffnen
CD76   20 E8 D4   JSR $D4E8        Pufferzeiger initialisieren
CD79   A8         TAY              und holen
CD7A   88         DEY              Zeiger auf vorhergehendes Zeichen
CD7B   C9 02      CMP #$02         mit Beginn des Datenbereichs vergleichen
CD7D   B0 02      BCS $CD81        ist Zeiger richtig gesetzt ?
CD7F   A0 01      LDY #$01         ja, Bytewert für aktuelle Pufferposition
CD81[1] A9 00     LDA #$00         Position des Pufferzeigers
CD83   20 C8 D4   JSR $D4C8        Pufferzeiger setzen
CD86   98         TYA              Position im Puffer
CD87   20 F1 CF   JSR $CFF1        Byte in Puffer schreiben
CD8A   8A         TXA              Puffernummer verdoppeln
CD8B   48         PHA              und merken
CD8C   20 64 D4   JSR $D464        Sektor auf Diskette schreiben
CD8F   68         PLA              Puffernummer wieder holen
CD90   AA         TAX              und setzen
CD91   20 AE FF   JSR $FFAE        Pufferzeiger neu setzen
CD94   4C 94 C1   JMP $C194        Rückmeldung bereitstellen und beenden
------------------------------------------------------------------------
[Vektor: FFEC]
Routine für U2-Befehl (vgl B-W); Sektor aus Puffer auf Diskette schreiben
CD97   20 6F CC   JSR $CC6F        Parameter aus Befehlsstring holen
CD9A   20 F2 CD   JSR $CDF2        Parameter prüfen und setzen
CD9D   20 64 D4   JSR $D464        Sektor auf Diskette schreiben
CDA0   4C 94 C1   JMP $C194        Rückmeldung bereitstellen und beenden
------------------------------------------------------------------------
[Einsprung über Routine CC1B]
Routine für Block-Execute-Befehl ('B-E'); Sektor lesen und ausführen
CDA3   20 58 F2   JSR $F258        keine Funktion (rts)
CDA6   20 36 CD   JSR $CD36        Sektor in Puffer einlesen
CDA9   A9 00      LDA #$00         Pufferadresse (Low-Byte) auf
CDAB   85 6F      STA $6F          Pufferanfang setzen
CDAD   A6 F9      LDX $F9          Puffernummer holen
CDAF   BD E0 FE   LDA $FEE0,X      High-Byte der Pufferadresse holen
CDB2   85 70      STA $70          und in Zeiger auf Pufferbeginn setzen
CDB4   20 BA CD   JSR $CDBA        Programm im Puffer starten
CDB7   4C 94 C1   JMP $C194        bei Rücksprung mit 'RTS' Befehl beenden
CDBA[1] 6C 6F 00  JMP ($006F)      über Zeiger in Puffer springen
------------------------------------------------------------------------
```

```
[Einsprung über Routine CC1B]
Routine für Block-Pointer-Befehl ('B-P'); Pufferzeiger setzen
CDBD  20 D2 CD   JSR $CDD2      Puffer belegen und Kanal öffnen
CDC0  A5 F9      LDA $F9        Puffernummer holen
CDC2  0A         ASL A          verdoppeln (da Pufferzeiger aus 2-Bytes)
CDC3  AA         TAX            und merken
CDC4  AD 86 02   LDA $0286      neue Position des Pufferzeigers holen
CDC7  95 99      STA $99,X      und als Low-Byte in Pufferzeiger setzen
CDC9  20 2F D1   JSR $D12F      Puffer- und Kanalnummer holen
CDCC  20 EE D3   JSR $D3EE      Byte aus aktueller Pufferposition holen
CDCF  4C 94 C1   JMP $C194      Rückmeldung bereitstellen und beenden
----------------------------------------------------------------------
[CDBD/CDF2]
Puffer belegen und Kanal eröffnen
CDD2  A6 D3      LDX $D3        Nummer des Parameters holen
CDD4  E6 D3      INC $D3        auf nächste Angabe setzen
CDD6  BD 85 02   LDA $0285,X    Kanalnummer aus Tabelle holen
CDD9  A8         TAY            und merken
CDDA  88         DEY            Kanalnummer um
CDDB  88         DEY            zwei verkleinern
CDDC  C0 0D      CPY #$0D       und mit Wert für Kanal 14 vergleichen
CDDE  90 05      BCC $CDE5      ist die Kanalnummer kleiner 15 ?
CDE0[1] A9 70    LDA #$70       nein, Fehlermeldung
CDE2  4C C8 C1   JMP $C1C8      '70 No Channel' ausgeben
CDE5[1] 85 83    STA $83        Kanalnummer als Sekundäradresse setzen
CDE7  20 EB D0   JSR $D0EB      und Kanal öffnen
CDEA  B0 F4      BCS $CDE0      war Kanal schon offen ?
CDEC  20 93 DF   JSR $DF93      nein, Puffernummer holen
CDEF  85 F9      STA $F9        und setzen
CDF1  60         RTS            zurück zur aufrufenden Routine
----------------------------------------------------------------------
[CD03/CD36/CD73/CD9A]
Parameter auf gültige Sektorangabe prüfen
CDF2  20 D2 CD   JSR $CDD2      Puffer belegen
CDF5  A6 D3      LDX $D3        Nummer des Parameters
CDF7  BD 85 02   LDA $0285,X    Byte aus Zwischenspeicher holen
CDFA  29 01      AND #$01       und Laufwerksnummer isolieren
CDFC  85 7F      STA $7F        als aktuelles Laufwerk übernehmen
CDFE  BD 87 02   LDA $0287,X    Nummer der gewünschten Spur
CE01  85 81      STA $81        setzen
CE03  BD 86 02   LDA $0286,X    Nummer des gewünschten Sektors
```

```
CE06  85 80     STA $80       übernehmen
CE08  20 5F D5  JSR $D55F     auf gültige Spur und Sektor prüfen
CE0B  4C 00 C1  JMP $C100     LED am aktuellen Laufwerk einschalten
-----------------------------------------------------------------------
[E255/E338/E436]
Record aus relativer Datei holen
CE0E  20 2C CE  JSR $CE2C     Zahl der Bytes bis zum Record errechnen
CE11  20 6E CE  JSR $CE6E     und Sektornummer des Records feststellen
CE14  A5 90     LDA $90       Rest der Division holen und als
CE16  85 D7     STA $D7       Pufferzeiger auf Start des Record setzen
CE18  20 71 CE  JSR $CE71     Side-Sektor der auf Rezord zeigt holen
CE1B  E6 D7     INC $D7       Pufferzeiger in pysikalischen
CE1D  E6 D7     INC $D7       Sektor um Verkettungsbytes korregieren
CE1F  A5 8B     LDA $8B       Nummer des Side-Sektors holen
CE21  85 D5     STA $D5       und merken
CE23  A5 90     LDA $90       Rest der Division holen und daraus
CE25  0A        ASL A         die Position des Sektorzeigers für den
CE26  18        CLC           Record im errechneten Side-Sektor
CE27  69 10     ADC #$10      bestimmen und
CE29  85 D6     STA $D6       merken
CE2B  60        RTS           zurück zur aufrufenden Routine
-----------------------------------------------------------------------
[CE0E]
Zahl der Bytes bis zum Record errechnen
CE2C  20 D9 CE  JSR $CED9     Zwischenspeicher löschen
CE2F  85 92     STA $92       Wert in Rechenregister 2
CE31  A6 82     LDX $82       Nummer des Kanals (Puffer) holen und
CE33  B5 B5     LDA $B5,X     dazugehörige Recordnummer (Low-Byte)
CE35  85 90     STA $90       bestimmen und übernehmen
CE37  B5 BB     LDA $BB,X     High-Byte der Recordnummer holen und
CE39  85 91     STA $91       übernehmen
CE3B  D0 04     BNE $CE41     ist Recordnummer größer 255 ?
CE3D  A5 90     LDA $90       nein, Low-Byte der Recordnummer holen
CE3F  F0 0B     BEQ $CE4C     ist Recordnummer gleich Null ?
CE41[1] A5 90   LDA $90       nein, Recordnummer (Low-Byte) holen
CE43  38        SEC           und um eins
CE44  E9 01     SBC #$01      verkleinern und neuen
CE46  85 90     STA $90       Wert übernehmen
CE48  B0 02     BCS $CE4C     ist Recordnummer kleiner 1  gewesen ?
CE4A  C6 91     DEC $91       ja, dann High-Byte um eins erniedrigen
CE4C[2] B5 C7   LDA $C7,X     Länge des Records holen
```

```
CE4E     85 6F      STA $6F       und merken
CE50[1]  46 6F      LSR $6F       auf geraden Wert prüfen
CE52     90 03      BCC $CE57     ist die Recordlänge gerade ?
CE54     20 ED CE   JSR $CEED     nein, Register 2 zu Register 1 addieren
CE57[1]  20 E5 CE   JSR $CEE5     Rechenregister mal 2
CE5A     A5 6F      LDA $6F       aktueller Record
CE5C     D0 F2      BNE $CE50     Bits einrechnen
CE5E     A5 D4      LDA $D4       Zeiger auf Position in Record
CE60     18         CLC           zu aktuellem
CE61     65 8B      ADC $8B       Rechenregister 1 dazuzählen
CE63     85 8B      STA $8B       Low-Byte wieder setzen
CE65     90 06      BCC $CE6D     ist ein Übertrag entstanden ?
CE67     E6 8C      INC $8C       ja, nächstes Byte korregieren
CE69     D0 02      BNE $CE6D     ist auch dabei ein Übertrag entstanden ?
CE6B     E6 8D      INC $8D       ja, höhstes Byte korregieren
CE6D[2]  60         RTS           zurück zur aufrufenden Routine
----------------------------------------------------------------------
[CE11]
Division des Rechenregisters durch 254 (Sektorlänge)
CE6E     A9 FE      LDA #$FE      Wert des Divisiors setzen (254)
CE70     2C         .byte $2C     zwei Bytes überspringen (Bit-Befehl)
----------------------------------------------------------------------
[CE18]
Division des Rechenregisters durch 120 (Recordeinträge in Side-Sektor)
CE71     A9 78      LDA #$78      Wert des Divisiors setzen (120)
CE73     85 6F      STA $6F       und merken
CE75     A2 03      LDX #$03      Zahl der Bytes pro Rechenregister
CE77[1]  B5 8F      LDA $8F,X     aktuellen Inhalt
CE79     48         PHA           retten
CE7A     B5 8A      LDA $8A,X     Bereich $88-$8A nach
CE7C     95 8F      STA $8F,X     Register 2 kopieren
CE7E     68         PLA           Inhalt von vorherigem Register 2 in
CE7F     95 8A      STA $8A,X     Bereich $88-$8A (Umtausch)
CE81     CA         DEX           Zeiger auf nächstes Byte
CE82     D0 F3      BNE $CE77     schon ganzes Register vertauscht ?
CE84     20 D9 CE   JSR $CED9     ja, Register 1 löschen
CE87[1]  A2 00      LDX #$00      Zähler initialisieren
CE89[1]  B5 90      LDA $90,X     Byte aus Register 2 holen
CE8B     95 8F      STA $8F,X     und um ein Byte vorverschieben
CE8D     E8         INX           Zeiger auf nächstes Byte
CE8E     E0 04      CPX #$04      mit Zahl der Registerbytes vergleichen
```

```
CE90   90 F7      BCC $CE89      schon ganzes Register verschoben ?
CE92   A9 00      LDA #$00       ja, höherwertiges Byte
CE94   85 92      STA $92        löschen
CE96   24 6F      BIT $6F        Divisor testen
CE98   30 09      BMI $CEA3      ist er größer 128
CE9A   06 8F      ASL $8F        nein, Bit0 aus niederwertigstem Teil von
CE9C   08         PHP            Register 2 ins Carry und merken
CE9D   46 8F      LSR $8F        Register wieder herstellen
CE9F   28         PLP            Carry wieder holen
CEA0   20 E6 CE   JSR $CEE6      und in Register 2 einschieben
CEA3 1 20 ED CE   JSR $CEED      Register 1 zu Register 2 addieren
CEA6   20 E5 CE   JSR $CEE5      Register 2 verdoppeln
CEA9   24 6F      BIT $6F        Divisor testen
CEAB   30 03      BMI $CEB0      ist er größer 128 ?
CEAD   20 E2 CE   JSR $CEE2      nein, Register 2 mal 4 nehmen
CEB0 1 A5 8F      LDA $8F        vorigen Wert
CEB2   18         CLC            in Rechenregister 2
CEB3   65 90      ADC $90        dazuaddieren
CEB5   85 90      STA $90        und wieder merken
CEB7   90 06      BCC $CEBF      ist ein Übertrag entstanden ?
CEB9   E6 91      INC $91        ja, 2. Byte des Registers korregieren
CEBB   D0 02      BNE $CEBF      ist auch dabei ein Übertrag enstanden ?
CEBD   E6 92      INC $92        ja, höchstes Byte des Registers setzen
CEBF 2 A5 92      LDA $92        und holen
CEC1   05 91      ORA $91        2. Byte einblenden
CEC3   D0 C2      BNE $CE87      sind beide Bytes 0 (Registerwert <256) ?
CEC5   A5 90      LDA $90        ja, niederwertigstes Registerbyte holen
CEC7   38         SEC            und davon den
CEC8   E5 6F      SBC $6F        Divisor abziehen
CECA   90 0C      BCC $CED8      enstand ein Übertrag ?
CECC   E6 8B      INC $8B        nein, Register 1 erhöhen
CECE   D0 06      BNE $CED6      Übertrag ?
CED0   E6 8C      INC $8C        2. Byte korregieren
CED2   D0 02      BNE $CED6      Übertrag ?
CED4   E6 8D      INC $8D        letztes Byte korregieren
CED6 2 85 90      STA $90        neuen Wert setzen
CED8 1 60         RTS            zurück zur aufrufenden Routine
-----------------------------------------------------------------
```

```
[CE2C/CE84]
Rechenregister 1 ($8B/$8C/$8D) löschen
CED9   A9 00      LDA #$00      Wert mit dem gelöscht
CEDB   85 8B      STA $8B       werden soll in
CEDD   85 8C      STA $8C       Rechenregister
CEDF   85 8D      STA $8D       übertragen
CEE1   60         RTS           zurück zur aufrufenden Routine
----------------------------------------------------------------
[CEAD]
Rechenregister 2 ($90/$91/$92) mal vier nehmen
CEE2   20 E5 CE   JSR $CEE5     Registerinhalt verdoppeln
----------------------------------------------------------------
[CE57/CEA6/CEE2/CEE6:CEA0]
Rechenregister 2 ($90/$91/$92) verdoppeln
CEE5   18         CLC           Einzuschiebender Wert =0
CEE6[1] 26 90     ROL $90       Wert in Register einschieben
CEE8   26 91      ROL $91       und gesamtes Register um
CEEA   26 92      ROL $92       eine Bitposition nach links schieben
CEEC   60         RTS           zurück zur aufrufenden Routine
----------------------------------------------------------------
[CE54/CEA3]
Rechenregister 2 ($90/$91/$92) zu Rechenregister 1 ($8B/$8C/$8D) addieren
CEED   18         CLC           Addition einleiten
CEEE   A2 FD      LDX #$FD      Zahl der Bytes der Register (neg. Wert)
CEF0[1] B5 8E     LDA $8E,X     Byte aus Register 1 holen
CEF2   75 93      ADC $93,X     Wert aus Register 2 holen und addieren
CEF4   95 8E      STA $8E,X     und Ergebnis in Register 1 speichern
CEF6   E8         INX           Zeiger auf nächste Ziffer setzen
CEF7   D0 F7      BNE $CEF0     bereits gesamtes Register addiert ?
CEF9   60         RTS           ja, zurück zur aufrufenden Routine
----------------------------------------------------------------
[CF17/EBBF]
Puffer-Kanal-Tabelle initialisieren
CEFA   A2 00      LDX #$00      Beginn bei Puffer 0
CEFC[1] 8A        TXA           Wert für Kanalzuweisung (0)
CEFD   95 FA      STA $FA,X     Kanalzuweisung des Puffers löschen
CEFF   E8         INX           nächsten Puffer anwählen
CF00   E0 04      CPX #$04      auf Nummer des größten Puffers prüfen
CF02   D0 F8      BNE $CEFC     schon alle Puffer bearbeitet ?
CF04   A9 06      LDA #$06      ja, Puffer 4 für Kanal 6
CF06   95 FA      STA $FA,X     belegen (BAM)
```

```
CF08  60          RTS           zurück zur aufrufenden Routine
--------------------------------------------------------------------

[CF1E/CF7B]
Kanalnummer in Puffer-Kanal-Tabelle prüfen
CF09  A0 04       LDY #$04      Zahl der Puffer
CF0B  A6 82       LDX $82       Nummer des gesuchten Kanals
CF0D[1] B9 FA 00  LDA $00FA,Y   zugeordnete Kanalnummer des Puffers
CF10  96 FA       STX $FA,Y     neue Nummer setzen
CF12  C5 82       CMP $82       alte Nummer mit neuer vergleichen
CF14  F0 07       BEQ $CF1D     sind beide identisch ?
CF16  88          DEY           nein, nächsten Puffer anwählen
CF17  30 E1       BMI $CEFA     war das der letzte Puffer ?
CF19  AA          TAX           nein, alte Kanalnummer
CF1A  4C 0D CF    JMP $CF0D     übernehmen und prüfen
CF1D[1] 60        RTS           zurück zur aufrufenden Routine
--------------------------------------------------------------------

[BF5A/D0B7/D0C0/D16A/D180/D18C/D1BB/DB2F/DB7D/DBA2/E04A/E05D/E072/E078]
[E18D/E19A/E19D/E2B9/E3B6/E3C8/E439/E451]
Puffer verwalten und zuweisen
CF1E  20 09 CF    JSR $CF09     Puffertabelle aktualisieren
CF21  20 B7 DF    JSR $DFB7     Status des gewählten Puffers holen
CF24  D0 46       BNE $CF6C     ist Puffer frei ?
CF26  20 D3 D1    JSR $D1D3     ja, zum Puffer gehörendes Laufw. setzen
CF29  20 8E D2    JSR $D28E     Puffer suchen
CF2C  30 48       BMI $CF76     wurde Puffer gefunden ?
CF2E  20 C2 DF    JSR $DFC2     ja, Puffer aktivieren
CF31  A5 80       LDA $80       Nummer der aktuellen Spur
CF33  48          PHA           merken
CF34  A5 81       LDA $81       Nummer des aktuellen Sektors
CF36  48          PHA           merken
CF37  A9 01       LDA #$01      Zeiger auf Position in Puffer
CF39  20 F6 D4    JSR $D4F6     ein Byte aus Puffer holen
CF3C  85 81       STA $81       und als Sektornummer merken
CF3E  A9 00       LDA #$00      Zeiger auf Position in Puffer
CF40  20 F6 D4    JSR $D4F6     Byte aus Puffer holen
CF43  85 80       STA $80       und als Spurnummer merken
CF45  F0 1F       BEQ $CF66     weiterer Sektor in Verkettung ?
CF47  20 25 D1    JSR $D125     ja, aktuellen Dateityp holen
CF4A  F0 0B       BEQ $CF57     gehört der Sektor zu einer REL-Datei ?
CF4C  20 AB DD    JSR $DDAB     nein, letzten Jobcode prüfen
CF4F  D0 06       BNE $CF57     war es ein Schreibvorgang ?
```

```
CF51    20 8C CF    JSR $CF8C     ja, Pufferzustand wechseln (ein/aus)
CF54    4C 5D CF    JMP $CF5D     und weitermachen
CF57[2] 20 8C CF    JSR $CF8C     Pufferzustand wechseln (aktiv/passiv)
CF5A    20 57 DE    JSR $DE57     Jobcode für 'Sektor lesen' setzen
CF5D[1] 68          PLA           aktuelle Sektornummer
CF5E    85 81       STA $81       wieder herstellen
CF60    68          PLA           aktuelle Spurnummer
CF61    85 80       STA $80       wieder herstellen
CF63    4C 6F CF    JMP $CF6F     und weitermachen
CF66[1] 68          PLA           aktuelle Sektornummer
CF67    85 81       STA $81       wieder herstellen
CF69    68          PLA           aktuelle Spurnummer
CF6A    85 80       STA $80       wieder herstellen
CF6C[1] 20 8C CF    JSR $CF8C     Pufferzustand wechseln (aktiv/passiv)
CF6F[1] 20 93 DF    JSR $DF93     Nummer des Puffers holen
CF72    AA          TAX           und merken
CF73    4C 99 D5    JMP $D599     und warten bis Job ausgeführt
CF76[2] A9 70       LDA #$70      Fehlermeldung
CF78    4C C8 C1    JMP $C1C8     '70 No Channel' ausgeben
-----------------------------------------------------------------------

[E325]
Nach freiem Puffer suchen
CF7B    20 09 CF    JSR $CF09     Puffertabelle aktualisieren
CF7E    20 B7 DF    JSR $DFB7     Nummer eines Puffers holen
CF81    D0 08       BNE $CF8B     ist Puffer frei ?
CF83    20 8E D2    JSR $D28E     nein, anderen Puffer wählen
CF86    30 EE       BMI $CF76     wurde ein anderer Puffer gefunden ?
CF88    20 C2 DF    JSR $DFC2     ja, Puffer aktivieren
CF8B[1] 60          RTS           zurück zur aufrufenden Routine
-----------------------------------------------------------------------

[CF51/CF57/CF6C]
Puffer von aktiv nach passiv umschalten und umgekehrt
CF8C    A6 82       LDX $82       Nummer des aktuellen Kanals
CF8E    B5 A7       LDA $A7,X     dazugehörigen Pufferstatus holen
CF90    49 80       EOR #$80      Flag für Puffer ein/aus wechseln
CF92    95 A7       STA $A7,X     und wieder zurückschreiben
CF94    B5 AE       LDA $AE,X     Nummer des 2. Puffers holen
CF96    49 80       EOR #$80      und umschalten
CF98    95 AE       STA $AE,X     neuen Wert in Tabelle schreiben
CF9A    60          RTS           zurück zur aufrufenden Routine
-----------------------------------------------------------------------
```

```
[C9D5/C9E7]
Bytes über internen Kanal in Puffer schreiben
CF9B   A2 12      LDX #$12      Nummer der Schreibkanals (18)
CF9D   86 83      STX $83       als aktuelle Sekundäradresse setzen
CF9F   20 07 D1   JSR $D107     Kanal suchen und öffnen
CFA2   20 00 C1   JSR $C100     LED am aktuellen Laufwerk einschalten
CFA5   20 25 D1   JSR $D125     Typ der dazugehörenden Datei holen
CFA8   90 05      BCC $CFAF     liegt eine Relative Datei vor ?
CFAA   A9 20       LDA #$20       ja, Flag für 'Datei nicht geschlossen'
CFAC   20 9D DD   JSR $DD9D     löschen
CFAF[1] A5 83     LDA $83       aktuelle Sekundäradresse holen
CFB1   C9 0F      CMP #$0F      mit Nummer für Kommandokanal vergleichen
CFB3   F0 23      BEQ $CFD8     ist Kommandokanal verlangt ?
CFB5   D0 08      BNE $CFBF     nein, immer Sprung nach $CBBF
-----------------------------------------------------------------------
[835C/EA48]
Byte in Datei schreiben
CFB7   A5 84      LDA $84       letzte Sekundäradresse
CFB9   29 8F      AND #$8F      Nummer des Kanals holen
CFBB   C9 0F      CMP #$0F      und auf Kommandokanal prüfen
CFBD   B0 19      BCS $CFD8     wurde Dateikanal angewählt
CFBF[1] 20 25 D1  JSR $D125     aktuellen Dateityp holen
CFC2   B0 05      BCS $CFC9     liegt 'REL' oder 'USR' vor ?
CFC4   A5 85      LDA $85       nein, aktuelles Datenbyte holen und
CFC6   4C 9D D1   JMP $D19D     in aktuellen Puffer schreiben
CFC9[1] D0 03     BNE $CFCE     ist Typ eine relative Datei ?
CFCB   4C AB E0   JMP $E0AB     ja, Byte in aktuellen Record übernehmen
CFCE[1] A5 85     LDA $85       aktuelles Datenbyte holen und
CFD0   20 F1 CF   JSR $CFF1     in Puffer schreiben
CFD3   A4 82      LDY $82       Nummer des aktuellen Kanals holen
CFD5   4C EE D3   JMP $D3EE     nächstes Byte zur Ausgabe holen
CFD8[2] A9 04     LDA #$04      höchste Kanalnummer (4)
CFDA   85 82      STA $82       als Nummer für Kommandokanal setzen
CFDC   20 E8 D4   JSR $D4E8     Pufferzeiger initialisieren
CFDF   C9 2A      CMP #$2A      auf Wert für Ende des Puffers prüfen
CFE1   F0 05      BEQ $CFE8     ist Puffer voll ?
CFE3   A5 85      LDA $85       nein, aktuelles Datenbyte holen
CFE5   20 F1 CF   JSR $CFF1     und in Puffer übertragen
CFE8[1] A5 F8     LDA $F8       Flag für letztes Byte (EOI) testen
CFEA   F0 01      BEQ $CFED     keine weiteren Daten ?
CFEC   60         RTS           ja, zurück zur aufrufenden Routine
```

```
CFED¹ EE 55 02   INC $0255        Befehlsmodusflag löschen
CFF0  60         RTS              zurück zur aufrufenden Routine
-----------------------------------------------------------------
[CD87/CFD0/CFE5/D19D/D1B0/D1B5/D4A8/D4AD/D4BB/D4C0/D4C5/D74D/D754/D75B]
[DB73/DB95/DB99/ECBE/ECC3/ECC8/ECCB/ECD1/ECD6/ECE7/ECEC/ECEF/ECFA/ED00]
[ED08/ED26/ED2C/ED3D/ED40/ED43/ED5E/CFFD:DD92]
Byte in aktuellen Puffer schreiben
CFF1  48         PHA              Byte merken
CFF2  20 93 DF   JSR $DF93        Nummer des Puffers holen
CFF5  10 06      BPL $CFFD        ist Puffer richtig zugeordnet ?
CFF7  68         PLA              nein, Stack korregieren
CFF8  A9 61      LDA #$61         Fehlermeldung
CFFA  4C C8 C1   JMP $C1C8        '61 File Not Open' ausgeben
CFFD² 0A         ASL A            Puffernummer verdoppeln
CFFE  AA         TAX              und merken
CFFF  68         PLA              Byte wieder holen
D000  81 99      STA ($99,X)      und in aktuellen Puffer schreiben
D002  F6 99      INC $99,X        Pufferzeiger auf nächstes Zeichen setzen
D004  60         RTS              zurück zur aufrufenden Routine
-----------------------------------------------------------------
[Einsprung über Routine C146]
Routine für Initialize-Befehl ('i')
D005  20 D1 C1   JSR $C1D1        Parameter holen
D008  20 42 D0   JSR $D042        BAM von Diskette einlesen
D00B  4C 94 C1   JMP $C194        Rückmeldung bereitstellen und beenden
-----------------------------------------------------------------
[C64C/D048]
aktuelles Laufwerk initialisieren
D00E  20 0F F1   JSR $F10F        Kanalnummer holen
D011  A8         TAY              und merken
D012  B6 A7      LDX $A7,Y        entsprechenden Pufferstatus holen
D014  E0 FF      CPX #$FF         mit Flagwert für 'belegt' vergleichen
D016  D0 14      BNE $D02C        ist Puffer frei ?
D018  48         PHA              ja, Kanalnummer merken
D019  20 8E D2   JSR $D28E        Puffer suchen und Zeiger setzen
D01C  AA         TAX              Nummer des Puffers holen
D01D  10 05      BPL $D024        wurde Puffer gefunden ?
D01F  A9 70      LDA #$70         nein, Fehlermeldung
D021  20 48 E6   JSR $E648        '70 No Channel' ausgeben
D024¹ 68         PLA              Kanalnummer wieder holen
D025  A8         TAY              und merken
```

```
D026  8A        TXA            Puffernummer holen
D027  09 80     ORA #$80       Flagwert für Puffer aktiv
D029  99 A7 00  STA $00A7,Y    in Kanal-Puffer-Tabelle schreiben
D02C  8A        TXA            Puffernummer holen
D02D  29 0F     AND #$0F       und Flags ausblenden
D02F  85 F9     STA $F9        aktuelle Puffernummer merken
D031  A2 00     LDX #$00       aktuelle Sektornummer
D033  86 81     STX $81        setzen
D035  AE 85 FE  LDX $FE85      Nummer der Directoryspur
D038  86 80     STX $80        als aktuelle Spurnummer setzen
D03A  20 D3 D6  JSR $D6D3      Spur und Sektor für Jobschleife setzen
D03D  A9 B0     LDA #$B0       Jobcode für 'Sektor suchen'
D03F  4C E5 A6  JMP $A5C5      Diskette initialisieren
-----------------------------------------------------------------------
[8FE1/907C/C666/D008/D828/E63E/ED87/EE46/EEB1]
BAM in Puffer lesen
D042  20 D1 F0  JSR $F0D1      Spurnummer der BAM löschen
D045  20 13 D3  JSR $D313      Kanäle des anderen Laufwerks schließen
D048  20 0E D0  JSR $D00E      Laufwerk initialisiern
D04B  A6 7F     LDX $7F        aktuelle Laufwerksnummer holen
D04D  A9 00     LDA #$00       und dazugehöriges Flag für
D04F  9D 51 02  STA $0251,X    'BAM gültig' setzen
D052  8A        TXA            Laufwerksnummer
D053  0A        ASL A          verdoppeln (da Zeiger für 2 Laufwerke)
D054  AA        TAX            und merken
D055  A5 16     LDA $16        erstes Zeichen der ID aus Blockheader
D057  95 12     STA $12,X      holen und merken
D059  A5 17     LDA $17        zweites Zeichen der ID aus Blockheader
D05B  95 13     STA $13,X      holen und übernehmen
D05D  20 67 A6  JSR $A667      BAM von Diskette lesen
D060  A5 F9     LDA $F9        Nummer des aktuellen Puffers
D062  0A        ASL A          und verdoppeln
D063  AA        TAX            (da Adresse aus zwei Bytes besteht)
D064  A9 02     LDA #$02       Pufferadresse Low-Byte
D066  95 99     STA $99,X      in Tabelle dem Puffer zuordnen
D068  A1 99     LDA ($99,X)    Byte aus Puffer holen
D06A  A6 7F     LDX $7F        aktuelles Laufwerk holen
D06C  9D 01 01  STA $0101,X    Byte als Formatkennzeichen speichern
D06F  A9 00     LDA #$00       Flags für Diskwechsel löschen und Flag
D071  4C 1D AA  JMP $AA1D      für Laufwerk bereit setzen
D074  EA        NOP            unbenutztes Byte
```

```
-------------------------------------------------------------------------
[A83B/AA22/EEF1]
Gesamtzahl der freien Blocks berechnen
D075  20 3A EF   JSR $EF3A      Pufferadresse in Zeiger $6D/$6E setzen
D078  A0 04      LDY #$04       Pufferzeiger auf Beginn der BAM setzen
D07A  A9 00      LDA #$00       Blockzähler
D07C  AA         TAX            initialisieren
D07D  18         CLC            Zahl der freien Blocks des Tracks
D07E  71 6D      ADC ($6D),Y    aus BAM holen und zu Zähler addieren
D080  90 01      BCC $D083      ist ein Ubertrag entstanden ?
D082  E8         INX            ja, High-Byte des Zählers erhöhen
D083  C8         INY            Pufferzeiger auf
D084  C8         INY            die Zahl der freien
D085  C8         INY            Blocks der nächsten Spur
D086  C8         INY            setzen und Sektor-Bitmuster überspringen
D087  C0 48      CPY #$48       Zeiger auf Position von Spur 18 prüfen
D089  F0 F8      BEQ $D083      steht der Zeiger auf Wert für Spur 18 ?
D08B  C0 90      CPY #$90       nein, auf Stelle der letzten Spur testen
D08D  D0 EE      BNE $D07D      freie Blöcke aller Spuren addiert ?
D08F  48         PHA            ja, Blockzähler (Low-Byte) merken
D090  8A         TXA            High-Byte des Blockzählers holen
D091  A6 7F      LDX $7F        Nummer des Laufwerks holen und freie
D093  9D FC 02   STA $02FC,X    Blocks des Laufwerks merken
D096  68         PLA            Low-Byte der freien Blocks holen
D097  4C 51 A9   JMP $A951      Zahl der freien 1571 Blocks berechnen
D09A  60         RTS            zurück zur aufrufenden Routine
-------------------------------------------------------------------------
[D0AF/DC57]
Sektor von Diskette in Puffer lesen
D09B  20 D0 D6   JSR $D6D0      Spur- und Sektornummer an Jobschleife
D09E  20 C3 D0   JSR $D0C3      Jobcode für 'Sektor lesen' übergeben
D0A1  20 99 D5   JSR $D599      warten bis Sektor in Puffer gelesen ist
D0A4  20 37 D1   JSR $D137      erstes Byte aus Puffer holen und als
D0A7  85 80      STA $80        Spur des nächsten Sektors merken
D0A9  20 37 D1   JSR $D137      nächstes Byte aus Puffer holen und als
D0AC  85 81      STA $81        Sektornummer des nächsten Sektors setzen
D0AE  60         RTS            zurück zur aufrufenden Routine
-------------------------------------------------------------------------
```

```
[E2CD]
angegebenen Sektor und Folgesektor einlesen
D0AF  20 9B D0  JSR $D09B     Sektor von Diskette lesen
D0B2  A5 80     LDA $80       Spurnummer des nächsten Sektors holen
D0B4  D0 01     BNE $D0B7     weiterer Sektor vorhanden ?
D0B6  60        RTS           nein, zurück zur aufrufenden Routine
D0B7[1] 20 1E CF  JSR $CF1E   noch einen Puffer belegen und
D0BA  20 D0 D6  JSR $D6D0     Parameter des nächsten Sektors setzen
D0BD  20 C3 D0  JSR $D0C3     auch nächsten in zweiten Puffer lesen
D0C0  4C 1E CF  JMP $CF1E     ersten Puffer wieder aktivieren
--------------------------------------------------------------------

[D09E/D0BD/D189]
Sektor von Diskette lesen
D0C3  A9 80     LDA #$80      Jobcode für 'Sektor lesen' festlegen
D0C5  D0 02     BNE $D0C9     immer Sprung nach $D0C9
--------------------------------------------------------------------

[D1B8/D4B0/DB9C]
Sektor auf Diskette schreiben
D0C7  A9 90     LDA #$90      Jobcode für 'Sektor schreiben' setzen
D0C9[1] 8D 4D 02  STA $024D   Jobcode merken
D0CC  20 93 DF  JSR $DF93     Nummer des aktuellen Puffers holen
D0CF  AA        TAX           und merken
D0D0  20 06 D5  JSR $D506     Spur- und Sektornummer überprüfen
D0D3  8A        TXA           Puffernummer wieder holen
D0D4  48        PHA           und retten
D0D5  0A        ASL A         Nummer verdoppeln
D0D6  AA        TAX           (da 2-Byte-Werte)
D0D7  A9 00     LDA #$00      Pufferadresse (Low-Byte)
D0D9  95 99     STA $99,X     zurücksetzen
D0DB  20 25 D1  JSR $D125     aktueller Dateityp holen
D0DE  C9 04     CMP #$04      auf Kennzeichen für SEQ-Datei prüfen
D0E0  B0 06     BCS $D0E8     gehört Sektor zu SEQ-Datei ?
D0E2  F6 B5     INC $B5,X     Zahl der belegten Blocks der Datei +1
D0E4  D0 02     BNE $D0E8     ist ein Übertrag entstanden ?
D0E6  F6 BB     INC $BB,X     ja, High-Byte des Blockzählers +1
D0E8[2] 68        PLA         Puffernummer wieder holen
D0E9  AA        TAX           und setzen
D0EA  60        RTS           zurück zur aufrufenden Routine
--------------------------------------------------------------------
```

```
[81EB/C6E2/C9C3/CB45/CDE7/D39B/D90E/DE36/E20F/E680/E90A]
Kanal zum Lesen öffnen
D0EB    A5 83      LDA $83        aktuelle Sekundäradresse holen
D0ED    C9 13      CMP #$13       und mit max. Sekundäradresse vergleichen
D0EF    90 02      BCC $D0F3      ist Adresse im erlaubten Bereich ?
D0F1    29 0F      AND #$0F       Sekundäradresse auf Bereich von 0 bis 15
D0F3[1] C9 0F      CMP #$0F       begrenzen und auf Kanal 15 prüfen
D0F5    D0 02      BNE $D0F9      ist der Kommandokanal angesprochen ?
D0F7    A9 10      LDA #$10       ja, Sekundäradresse 16 (Fehlerkanal)
D0F9[1] AA         TAX            setzen
D0FA    38         SEC            Flag für 'Kanal nicht zum Lesen'
D0FB    BD 2B 02   LDA $022B,X    Status des Kanals prüfen
D0FE    30 06      BMI $D106      ist Kanal im Lesemodus ?
D100    29 0F      AND #$0F       ja, interne Kanalnummer herstellen
D102    85 82      STA $82        als aktuelle Kanalnummer setzen
D104    AA         TAX            und merken
D105    18         CLC            Flag für 'Kanal eröffnet' setzen
D106[1] 60         RTS            zurück zur aufrufenden Routine
------------------------------------------------------------------------
[8343/CA63/CF9F/DB1B/DC43/E688/EA2F]
Kanal suchen und öffnen
D107    A5 83      LDA $83        aktuelle Sekundäradresse holen
D109    C9 13      CMP #$13       und mit maximalem Wert (19) vergleichen
D10B    90 02      BCC $D10F      ist Adresse im erlaubten Bereich ?
D10D    29 0F      AND #$0F       nein, dann auf Bereich umrechnen
D10F[1] AA         TAX            und merken
D110    BD 2B 02   LDA $022B,X    entsprechenden Kanalstatus holen
D113    A8         TAY            und merken
D114    0A         ASL A          Status zum testen von Bit 6/7 richten
D115    90 0A      BCC $D121      ist Flag für Schreiben gesetzt ?
D117    30 0A      BMI $D123      ja, ist Flag für Lesen gesetzt ?
D119[1] 98         TYA            nein, Kanalstatus wieder holen
D11A    29 0F      AND #$0F       reine Kanalnummer herstellen
D11C    85 82      STA $82        als aktuellen Kanal setzen
D11E    AA         TAX            und merken
D11F    18         CLC            Flag für 'Kanal geöffnet' setzen
D120    60         RTS            zurück zur aufrufenden Routine
D121[1] 30 F6      BMI $D119      ist Flag für Lesen gesetzt ?
D123[1] 38         SEC            ja, Flag für 'Kanal nur lesen' setzen
D124    60         RTS            zurück zur aufrufenden Routine
------------------------------------------------------------------------
```

[C979/C9C6/C9E2/CA2B/CA48/CF47/CFA5/CFBF/D0DB/D3AC/D3C0/DB10/DBDE]
[E21E/E68E]
Aktuellen Dateityp holen

```
D125  A6 82     LDX $82        Nummer des aktuellen Kanals holen
D127  B5 EC     LDA $EC,X      und dazugehörigen Dateityp holen
D129  4A        LSR A          Laufwerksnummer ignorieren
D12A  29 07     AND #$07       Kennzeichen für Dateityp herstellen
D12C  C9 04     CMP #$04       mit Kode für REL-Datei vergleichen
D12E  60        RTS            zurück zur aufrufenden Routine
```

[CD3C/CDC9/D137/D3DE/E01D/E127/E138/E156]
Kanal- und entsprechende Puffernummer holen und setzen

```
D12F  20 93 DF  JSR $DF93      Nummer des aktuellen Puffers holen
D132  0A        ASL A          und verdoppeln
D133  AA        TAX            als Zeiger auf 2-Byte-Werte merken
D134  A4 82     LDY $82        als aktuelle Kanalnummer speichern
D136  60        RTS            zurück zur aufrufenden Routine
```

[D0A4/D0A9/D156/D172/D17B/D192/D433/DAAA/DE9A/DE9F/ED67/EDF3/EDF8]
Byte aus aktuellem Puffer holen

```
D137  20 2F D1  JSR $D12F      Kanal- und Puffernummer setzen
D13A  B9 44 02  LDA $0244,Y    Zeiger auf Pufferende
D13D  F0 12     BEQ $D151      ist letztes Byte des Puffers gelesen ?
D13F  A1 99     LDA ($99,X)    nein, Byte aus Puffer holen
D141  48        PHA            und merken
D142  B5 99     LDA $99,X      Pufferzeiger (Low-Byte) holen
D144  D9 44 02  CMP $0244,Y    und auf logisches Pufferende prüfen
D147  D0 04     BNE $D14D      Ende des gültigen Puffers erreicht ?
D149  A9 FF     LDA #$FF       ja, Pufferzeiger auf physikalisches
D14B  95 99     STA $99,X      Pufferende setzen
D14D[1] 68      PLA            und Datenbyte wieder holen
D14E  F6 99     INC $99,X      Pufferzeiger auf Pufferanfang setzen
D150  60        RTS            zurück zur aufrufenden Routine
D151[1] A1 99   LDA ($99,X)    letztes Byte aus Puffer holen und
D153  F6 99     INC $99,X      Pufferzeiger wieder auf Anfang setzen
D155  60        RTS            zurück zur aufrufenden Routine
```

[C899/C89E/D400/D45C/DCA9]
Byte aus Datei holen

```
D156  20 37 D1  JSR $D137      Byte aus Puffer holen
D159  D0 36     BNE $D191      war das das letzte Byte des Puffers ?
```

```
D15B  85 85     STA $85       ja, Datenbyte merken
D15D  B9 44 02  LDA $0244,Y   Zeiger auf gültigen Pufferbereich holen
D160  F0 08     BEQ $D16A     ist das physikalische Ende erreicht ?
D162  A9 80     LDA #$80      nein, Flag für 'Lesen'
D164  99 F2 00  STA $00F2,Y   in Kanalstatustabelle setzen
D167  A5 85     LDA $85       Datenbyte wieder holen
D169  60        RTS           zurück zur aufrufenden Routine
D16A[1] 20 1E CF  JSR $CF1E   logisch nächsten Sektor einlesen
D16D  A9 00     LDA #$00      Pufferzeiger
D16F  20 C8 D4  JSR $D4C8     zurücksetzen
D172  20 37 D1  JSR $D137     erstes Byte aus Sektor holen und auf
D175  C9 00     CMP #$00      Kennzeichen für 'letzter Sektor' prüfen
D177  F0 19     BEQ $D192     kein weiterer Sektor vorhanden ?
D179  85 80     STA $80       nein, Spurnummer des nächsten Sektors
D17B  20 37 D1  JSR $D137     zweites Byte aus Sektor holen
D17E  85 81     STA $81       und als Sektornummer speichern
D180  20 1E CF  JSR $CF1E     noch ein Puffer belegen
D183  20 D3 D1  JSR $D1D3     Puffer- und Laufwerksnummer setzen
D186  20 D0 D6  JSR $D6D0     Spur- und Sektornummer an Jobschleife
D189  20 C3 D0  JSR $D0C3     Sektor in Puffer lesen
D18C  20 1E CF  JSR $CF1E     auf vorherigen Puffer zurückschalten
D18F  A5 85     LDA $85       Datenbyte wieder holen
D191[1] 60      RTS           zurück zur aufrufenden Routine
D192[1] 20 37 D1  JSR $D137   Byte aus Puffer holen
D195  A4 82     LDY $82       Nummer des aktuellen Kanals holen
D197  99 44 02  STA $0244,Y   Zahl der zu übertragenden Bytes setzen
D19A  A5 85     LDA $85       Datenbyte wieder holen
D19C  60        RTS           zurück zur aufrufenden Routine
----------------------------------------------------------------------
[CFC6/D1A3:DA3D]
Byte in Datei schreiben
D19D  20 F1 CF  JSR $CFF1     Datenbyte in Puffer schreiben
D1A0  F0 01     BEQ $D1A3     ist Puffer jetzt voll ?
D1A2  60        RTS           nein, zurück zur aufrufenden Routine
D1A3[2] 20 D3 D1  JSR $D1D3   Puffer- und Laufwerksnummer setzen
D1A6  20 1E F1  JSR $F11E     nächsten freien Sektor aus BAM holen
D1A9  A9 00     LDA #$00      Pufferzeiger auf Verkettungsbytes
D1AB  20 C8 D4  JSR $D4C8     des Sektors setzen
D1AE  A5 80     LDA $80       Spurnummer des nächsten Sektors in
D1B0  20 F1 CF  JSR $CFF1     Verkettungsbytes des Sektors schreiben
D1B3  A5 81     LDA $81       Nummer des nächsten Sektors in
```

```
D1B5  20 F1 CF   JSR $CFF1      Verkettungsbytes des Sektors schreiben
D1B8  20 C7 D0   JSR $D0C7      Sektor auf Diskette schreiben
D1BB  20 1E CF   JSR $CF1E      auf nächsten Puffer wechseln
D1BE  20 D0 D6   JSR $D6D0      Spur- und Sektornummer für Job setzen
D1C1  A9 02      LDA #$02       Pufferzeiger auf Start des Datenbereichs
D1C3  4C C8 D4   JMP $D4C8      setzen
---------------------------------------------------------------------
[C623]
aktuellen Pufferzeiger auf nächstes Zeichen setzen
D1C6  85 6F      STA $6F        neue Position des Zeigers merken
D1C8  20 E8 D4   JSR $D4E8      Zeiger des aktiven Puffers setzen
D1CB  18         CLC            und zu dem neuen
D1CC  65 6F      ADC $6F        Zeigerwert addieren
D1CE  95 99      STA $99,X      neuen Wert in Low-Byte des Zeigers
D1D0  85 94      STA $94        und Directorypufferzeiger übernehmen
D1D2  60         RTS            zurück zur aufrufenden Routine
---------------------------------------------------------------------
[CA53/CA66/CF26/D183/D1A3/E03C/E31C]
Nummer des dem Puffer zugeordneten Laufwerks holen
D1D3  20 93 DF   JSR $DF93      Nummer des Puffers feststellen
D1D6  AA         TAX            und merken
D1D7  BD 5B 02   LDA $025B,X    entsprechenden Jobcode aus Tabelle holen
D1DA  29 01      AND #$01       und daraus Laufwerksnummer herstellen
D1DC  85 7F      STA $7F        und als aktuelles Laufwerk speichern
D1DE  60         RTS            zurück zur aufrufenden Routine
---------------------------------------------------------------------
[DCDF]
Schreibkanal und Puffer suchen
D1DF  38         SEC            Flag für Schreiben setzen
D1E0  B0 01      BCS $D1E3      immer Sprung nach $D1E3
---------------------------------------------------------------------
[91D0/CB9A/CBDF/DC48/ECA4/D20F:DC7E,DD13]
Lesekanal und Puffer suchen
D1E2  18         CLC            Flag für Lesen setzen
D1E3[1] 08       PHP            Flag merken
D1E4  85 6F      STA $6F        Zahl der zu suchenden Puffer
D1E6  20 27 D2   JSR $D227      alle Kanäle löschen
D1E9  20 7F D3   JSR $D37F      nächsten freien Kanal suchen und belegen
D1EC  85 82      STA $82        Nummer des Kanals merken
D1EE  A6 83      LDX $83        Sekundäradresse holen
D1F0  28         PLP            Flag für Lesen/Schreiben wieder holen
```

```
D1F1    90 02      BCC $D1F5      soll Lesekanal geöffnet werden ?
D1F3    09 80      ORA #$80       nein, Flag für 'Schreiben' setzen
D1F5[1] 9D 2B 02   STA $022B,X    und in Statustabelle schreiben
D1F8    29 3F      AND #$3F       Nummer des internen Kanals herstellen
D1FA    A8         TAY            und merken
D1FB    A9 FF      LDA #$FF       zugehörige Puffer
D1FD    99 A7 00   STA $00A7,Y    eins und
D200    99 AE 00   STA $00AE,Y    zwei freigeben
D203    99 CD 00   STA $00CD,Y    dritten Puffer freigeben
D206    C6 6F      DEC $6F        Zahl der zu suchenden Puffer erniedrigen
D208    30 1C      BMI $D226      genügend Puffer gefunden ?
D20A    20 8E D2   JSR $D28E      nein, freien Puffer suchen
D20D    10 08      BPL $D217      wurde ein Puffer gefunden ?
D20F[3] 20 5A D2   JSR $D25A      nein, Puffer freigeben
D212    A9 70      LDA #$70       Fehlermeldung
D214    4C C8 C1   JMP $C1C8      '70 No Channel' ausgeben
D217[1] 99 A7 00   STA $00A7,Y    Nummer des Puffers in Belegungstabelle
D21A    C6 6F      DEC $6F        Zahl der zu suchenden Puffer erniedrigen
D21C    30 08      BMI $D226      genügend Puffer gefunden ?
D21E    20 8E D2   JSR $D28E      nein, nächsten Puffer suchen
D221    30 EC      BMI $D20F      wurde ein freier Puffer gefunden
D223    99 AE 00   STA $00AE,Y    ja, Puffernummer merken
D226[2] 60         RTS            zurück zur aufrufenden Routine
---------------------------------------------------------------------------
[C8AA/D1E6/D30B/D331/D4DE/D4E5/DACE/DB29/DB5F/E695/EE01]
Kanal freigeben
D227    A5 83      LDA $83        aktuelle Sekundäradresse holen und
D229    C9 0F      CMP #$0F       mit Wert für Kommandokanal vergleichen
D22B    D0 01      BNE $D22E      ist Kanal 15 aktiv ?
D22D    60         RTS            ja, zurück zur aufrufenden Routine
D22E[1] A6 83      LDX $83        aktuelle Sekundäradresse holen
D230    BD 2B 02   LDA $022B,X    dazugehörigen Kanalstatus feststellen
D233    C9 FF      CMP #$FF       auf Wert für 'Kanal unbenutzt' prüfen
D235    F0 22      BEQ $D259      ist Kanal frei ?
D237    29 3F      AND #$3F       nein, Kanalnummer herstellen
D239    85 82      STA $82        und merken
D23B    A9 FF      LDA #$FF       Flagwert für 'Kanal und Puffer frei'
D23D    9D 2B 02   STA $022B,X    in Kanaltabelle speichern
D240    A6 82      LDX $82        aktuelle Kanalnummer wieder holen
D242    A9 00      LDA #$00       Flags in
D244    95 F2      STA $F2,X      Kanaltabelle für Kanalstatus löschen
```

```
D246  20 5A D2   JSR $D25A       dazugehörigen Puffer freigeben
D249  A6 82      LDX $82         Nummer des aktuellen Kanals
D24B  A9 01      LDA #$01        Bitflag für 'Kanal frei'
D24D  CA         DEX             Nummer des Kanals erniedrigen
D24E  30 03      BMI $D253       ist Flag an richtiger Position ?
D250  0A         ASL A           nein, Bitflag in Bitmuster einschieben
D251  D0 FA      BNE $D24D       immer Sprung nach $D24D
D253  0D 56 02   ORA $0256       Flag in Bitverzeichnis der belegten
D256  8D 56 02   STA $0256       Kanäle schreiben
D259  60         RTS             zurück zur aufrufenden Routine
--------------------------------------------------------------------------
[D20F/D246]
Puffer und zugehörigen Kanal freigeben
D25A  A6 82      LDX $82         aktuelle Kanalnummer holen
D25C  B5 A7      LDA $A7,X       und Puffernummer des Kanals feststellen
D25E  C9 FF      CMP #$FF        mit Wert für 'Puffer frei' vergleichen
D260  F0 09      BEQ $D26B       ist Puffer dem Kanal zugeordnet ?
D262  48         PHA             ja, Nummer des Puffers merken
D263  A9 FF      LDA #$FF        und Puffer in
D265  95 A7      STA $A7,X       Puffertabelle freigeben
D267  68         PLA             Puffernummer wieder holen
D268  20 F3 D2   JSR $D2F3       Pufferbelegung freigeben
D26B  A6 82      LDX $82         Nummer des aktuellen Kanals
D26D  B5 AE      LDA $AE,X       Nummer des entsprechenden Puffers holen
D26F  C9 FF      CMP #$FF        und auf Wert für 'nicht belegt' prüfen
D271  F0 09      BEQ $D27C       ist der Puffer frei ?
D273  48         PHA             nein, Puffernummer merken
D274  A9 FF      LDA #$FF        Puffer des Kanals
D276  95 AE      STA $AE,X       freigeben
D278  68         PLA             und aktuelle Puffernummer wieder holen
D279  20 F3 D2   JSR $D2F3       Puffer in Belegungstabelle freigeben
D27C  A6 82      LDX $82         Nummer des aktuellen Kanals holen
D27E  B5 CD      LDA $CD,X       und entsprechende Puffernummer holen
D280  C9 FF      CMP #$FF        mit Wert für Puffer inaktiv vergleichen
D282  F0 09      BEQ $D28D       ist der Puffer benutzt ?
D284  48         PHA             ja, Nummer des Puffers merken
D285  A9 FF      LDA #$FF        und Zuordung des Puffers zum
D287  95 CD      STA $CD,X       aktuellen Kanal löschen
D289  68         PLA             Puffernummer wieder holen
D28A  20 F3 D2   JSR $D2F3       und Puffer in Belegungstabelle freigeben
D28D  60         RTS             zurück zur aufrufenden Routine
```

```
----------------------------------------------------------------------
[CF29/CF83/D019/D20A/D21E/DC79/DD0E/F0E7]
Puffer suchen
D28E  98          TYA           Nummer des Puffers holen
D28F  48          PHA           und merken
D290  A0 01       LDY #$01      einen freien
D292  20 BA D2    JSR $D2BA     Puffer suchen
D295  10 0C       BPL $D2A3     wurde ein Puffer gefunden ?
D297  88          DEY           nein, Puffernummer auf nächsten Puffer
D298  20 BA D2    JSR $D2BA     setzen und nochmal Puffer suchen
D29B  10 06       BPL $D2A3     wurde jetzt ein Puffer gefunden ?
D29D  20 39 D3    JSR $D339     nein, freien Puffer holen
D2A0  AA          TAX           Puffernummer merken
D2A1  30 13       BMI $D2B6     wurde ein Puffer gefunden ?
D2A3  B5 00       LDA $00,X     ja, letzten Jobcode des Puffers holen
D2A5  30 FC       BMI $D2A3     ist Job schon fertig ausgeführt ?
D2A7  A5 7F       LDA $7F       ja, aktuelle Laufwerksnummer holen
D2A9  95 00       STA $00,X     Rückmeldung der Jobschleife entfernen
D2AB  9D 5B 02    STA $025B,X   und Speicher für letzten Jobcode löschen
D2AE  8A          TXA           Nummer des Puffers holen
D2AF  0A          ASL A         und verdoppeln (da nachfolgende Adressen
D2B0  A8          TAY           aus Zwei-Byte-Werten bestehen)
D2B1  A9 02       LDA #$02      Pufferzeiger auf Start des Datenbereichs
D2B3  99 99 00    STA $0099,Y   Zeiger für Puffer neu setzen
D2B6  68          PLA           Nummer des Puffers wieder herstellen
D2B7  A8          TAY           und merken
D2B8  8A          TXA           Nummer des gefundenen Puffers setzen
D2B9  60          RTS           zurück zur aufrufenden Routine
----------------------------------------------------------------------
[D292/D298]
freien Puffer suchen
D2BA  A2 07       LDX #$07      Zahl der Bits pro Byte -1 (da 'BPL')
D2BC  B9 4F 02    LDA $024F,Y   Bitmuster der Belegungstabelle holen
D2BF  3D E9 EF    AND $EFE9,X   entsprechendes Bit des Puffers holen
D2C2  F0 04       BEQ $D2C8     ist der Puffer belegt
D2C4  CA          DEX           ja, Zähler für Puffer auf nächstes Bit
D2C5  10 F5       BPL $D2BC     schon alle Bits getestet ?
D2C7  60          RTS           ja, zurück zur aufrufenden Routine
D2C8  B9 4F 02    LDA $024F,Y   Orginalbyte der Belegungstabelle holen
D2CB  5D E9 EF    EOR $EFE9,X   und entsprechendes Bit des Puffers
D2CE  99 4F 02    STA $024F,Y   setzen und Byte wieder zurückschreiben
```

```
D2D1  8A        TXA          Nummer des gefundenen Puffers holen
D2D2  88        DEY          Zeiger auf nächste Verzeichnisbyte
D2D3  30 03     BMI $D2D8    schon alle beide bearbeitet ?
D2D5  18        CLC          nein, neue
D2D6  69 08     ADC #$08     Puffernummer berechnen
D2D8[1] AA      TAX          Puffernummer als Kanalnummer merken
D2D9[2] 60      RTS          zurück zur aufrufenden Routine
------------------------------------------------------------------
[E2BC/E2BF]
alle inaktiven Puffer freigeben
D2DA  A6 82     LDX $82      Nummer des aktuellen Kanals holen
D2DC  B5 A7     LDA $A7,X    und zugehörigen Puffer feststellen
D2DE  30 09     BMI $D2E9    ist Puffer belegt ?
D2E0  8A        TXA          ja,, Nummer des Kanals wieder holen
D2E1  18        CLC          und für zweiten Puffer
D2E2  69 07     ADC #$07     umrechnen
D2E4  AA        TAX          und merken
D2E5  B5 A7     LDA $A7,X    zugehörige Puffernummer holen
D2E7  10 F0     BPL $D2D9    ist Puffer belegt ?
D2E9[1] C9 FF   CMP #$FF     nein, auf Wert für 'Puffer frei' prüfen
D2EB  F0 EC     BEQ $D2D9    ist der Puffer als frei gekennzeichnet ?
D2ED  48        PHA          nein, Puffernummer merken
D2EE  A9 FF     LDA #$FF     Wert für 'Puffer frei' für
D2F0  95 A7     STA $A7,X    den aktuellen Kanal setzen
D2F2  68        PLA          und Puffernummer wieder holen
------------------------------------------------------------------
[D268/D279/D28A]
Puffer in Bitverzeichnis freigeben
D2F3  29 0F     AND #$0F     reiner Puffernummer
D2F5  A8        TAY          herstellen und
D2F6  C8        INY          merken
D2F7  A2 10     LDX #$10     Zahl der Puffer
D2F9[1] 6E 50 02 ROR $0250   Bitverzeichnis der Pufferbelegung
D2FC  6E 4F 02  ROR $024F    bis zum Bit des Puffers verschieben
D2FF  88        DEY          Zeiger auf nächsten Puffer
D300  D0 01     BNE $D303    noch ein Puffer weiter ?
D302  18        CLC          nein, Flag für 'Puffer frei' setzen
D303[1] CA      DEX          Bitverzeichnis wieder herstellen
D304  10 F3     BPL $D2F9    sind Bits wieder in Ausgangsposition ?
D306  60        RTS          ja, zurück zur aufrufenden Routine
------------------------------------------------------------------
```

```
[84D8/8C64/EE36]
Kanäle 0 bis 14 schließen
D307    A9 0E      LDA #$0E     Zähler für Kanalnummer
D309    85 83      STA $83      als aktuelle Sekundäradresse setzen
D30B[1] 20 27 D2   JSR $D227    und Kanal schließen
D30E    C6 83      DEC $83      Zähler auf nächste Kanalnummer setzen
D310    D0 F9      BNE $D30B    schon alle Kanäle geschlossen ?
D312    60         RTS          ja, zurück zur aufrufenden Routine
--------------------------------------------------------------------
[D045/EC55/EC66]
alle Kanäle des aktuellen Laufwerks freigeben
D313    A9 0E      LDA #$0E     Zähler für Kanalnummer
D315    85 83      STA $83      Kanalnummer als aktuelle Sekundäradresse
D317[1] A6 83      LDX $83      merken und setzen
D319    BD 2B 02   LDA $022B,X  dazugehörigen Status holen
D31C    C9 FF      CMP #$FF     und auf Wert für 'Kanal frei' prüfen
D31E    F0 14      BEQ $D334    ist Kanal belegt ?
D320    29 3F      AND #$3F     ja, reine Kanalnummer herstellen
D322    85 82      STA $82      und speichern
D324    20 93 DF   JSR $DF93    Nummer des Puffers holen
D327    AA         TAX          und merken
D328    BD 5B 02   LDA $025B,X  Jobcode für Puffer holen und daraus
D32B    29 01      AND #$01     Laufwerksangabe isolieren
D32D    C5 7F      CMP $7F      auf Wert des aktuellen Laufwerks prüfen
D32F    D0 03      BNE $D334    gehört Kanal zum anderen Laufwerk ?
D331    20 27 D2   JSR $D227    nein, Kanal freigeben
D334[2] C6 83      DEC $83      Zähler für Kanäle auf nächsten Kanal
D336    10 DF      BPL $D317    schon alle Kanäle bearbeitet ?
D338    60         RTS          ja, zurück zur aufrufenden Routine
--------------------------------------------------------------------
[D29D]
freien Puffer holen
D339    A5 6F      LDA $6F      Nummer des Kanals holen
D33B    48         PHA          und merken
D33C    A0 00      LDY #$00     Zähler für Kanalnummer auf Startwert
D33E[1] B6 FA      LDX $FA,Y    Nummer des Kanals holen
D340    B5 A7      LDA $A7,X    Nummer des zugeordneten Puffers holen
D342    10 04      BPL $D348    ist Puffer verwendet ?
D344    C9 FF      CMP #$FF     nein, auf Wert für 'Puffer frei' prüfen
D346    D0 16      BNE $D35E    ist Puffer frei ?
D348[2] 8A         TXA          ja, Kanalnummer wieder holen
```

```
D349    18        CLC            und für Zugriff auf
D34A    69 07     ADC #$07       zweiten Puffer umrechen
D34C    AA        TAX            und merken
D34D    B5 A7     LDA $A7,X      entsprechende Puffernummer holen
D34F    10 04     BPL $D355      ist Puffer belegt ?
D351    C9 FF     CMP #$FF       nein, auf Wert für 'Puffer frei' prüfen
D353    D0 09     BNE $D35E      ist Puffer frei ?
D355[2] C8        INY            ja, nächsten Kanal anwählen
D356    C0 05     CPY #$05       mit maximaler Kanalzahl vergleichen
D358    90 E4     BCC $D33E      schon alle Kanäle bearbeitet ?
D35A    A2 FF     LDX #$FF       Flagwert für Fehler
D35C    D0 1C     BNE $D37A      immer Sprung nach $D37A
D35E[2] 86 6F     STX $6F        Nummer des Kanals setzen
D360    29 3F     AND #$3F       und daraus Puffernummer herstellen
D362    AA        TAX            und merken
D363[1] B5 00     LDA $00,X      Jobcode des Puffers holen
D365    30 FC     BMI $D363      ist Job noch in Gang ?
D367    C9 02     CMP #$02       nein, Rückmeldung auf 'Ok'-Wert prüfen
D369    90 08     BCC $D373      ist Job fehlerfrei ausgeführt worden ?
D36B    A6 6F     LDX $6F        nein, Nummer des Kanals holen
D36D    E0 07     CPX #$07       und auf Wert für maximale Nummer testen
D36F    90 D7     BCC $D348      ist Kanalnummer im erlaubten Bereich ?
D371    B0 E2     BCS $D355      nein, immer Sprung nach $D355
D373[1] A4 6F     LDY $6F        Nummer des Kanals holen
D375    A9 FF     LDA #$FF       und Puffer in Pufferzuordungstabelle
D377    99 A7 00  STA $00A7,Y    als frei kennzeichnen
D37A[1] 68        PLA            Einsprungskanalnummer wieder holen
D37B    85 6F     STA $6F        und wieder zurücksetzen
D37D    8A        TXA            Puffernummer übergeben
D37E    60        RTS            zurück zur aufrufenden Routine
----------------------------------------------------------------------------
[D1E9]
freien Kanal suchen und belegen
D37F    A0 00     LDY #$00       Zeiger initialisieren
D381    A9 01     LDA #$01       Bit des zu testenden Kanals
D383[1] 2C 56 02  BIT $0256      Bit in Kanalverzeichnis prüfen
D386    D0 09     BNE $D391      ist Kanal frei ?
D388    C8        INY            nein, nächsten Kanal anwählen
D389    0A        ASL A          Bit auf Posistion des nächsten Kanals
D38A    D0 F7     BNE $D383      schon alle Kanäle geprüft ?
D38C    A9 70     LDA #$70       ja, Fehlermeldung
```

```
D38E  4C C8 C1   JMP $C1C8     '70 No Channel' ausgeben
D391  49 FF      EOR #$FF      Bitflag für 'Kanal frei' invertieren
D393  2D 56 02   AND $0256     und in Flagbyte einblenden
D396  8D 56 02   STA $0256     Kanal belegen
D399  98         TYA           Nummer des Kanals holen
D39A  60         RTS           zurück zur aufrufenden Routine
-----------------------------------------------------------------
[CA39]
Byte von Kanal holen
D39B  20 EB D0   JSR $D0EB     Lesekanal eröffnen
D39E  20 00 C1   JSR $C100     LED am aktuellen Laufwerk einschalten
D3A1  20 AA D3   JSR $D3AA     Byte über Kanal auslesen
D3A4  A6 82      LDX $82       Nummer des Kanals feststellen
D3A6  BD 3E 02   LDA $023E,X   und entsprechendes Datenbyte holen
D3A9  60         RTS           zurück zur aufrufenden Routine
-----------------------------------------------------------------
[82BD/D3A1/E992]
Byte aus Datei lesen
D3AA  A6 82      LDX $82       Nummer des Kanals holen
D3AC  20 25 D1   JSR $D125     Dateityp feststellen
D3AF  D0 03      BNE $D3B4     ist Datei eine Relative Datei ?
D3B1  4C 20 E1   JMP $E120     ja, Routine für REL-Datei
D3B4  A5 83      LDA $83       Sekundäradresse holen
D3B6  C9 0F      CMP #$0F      und mit Kommandokanal vergleichen (15)
D3B8  F0 5A      BEQ $D414     liegt Kommandokanal vor ?
D3BA  B5 F2      LDA $F2,X     nein, Status des Kanals holen
D3BC  29 08      AND #$08      und Flag für EOI testen
D3BE  D0 13      BNE $D3D3     wurde letztes Byte übertragen ?
D3C0  20 25 D1   JSR $D125     ja, Typ der Datei feststellen und
D3C3  C9 07      CMP #$07      mit Wert für Direktzugriff vergleichen
D3C5  D0 07      BNE $D3CE     wurde ein Direktzugriffkanal eröffnet ?
D3C7  A9 89      LDA #$89      ja, Flagwert für Direktzugriff
D3C9  95 F2      STA $F2,X     als Kanalstatus übernehmen
D3CB  4C DE D3   JMP $D3DE     Byte aus Puffer holen
D3CE  A9 00      LDA #$00      Flagwert für EOI aufgetreten; Kanal
D3D0  95 F2      STA $F2,X     schließen und Belegung löschen
D3D2  60         RTS           zurück zur aufrufenden Routine
D3D3  A5 83      LDA $83       aktuelle Sekundäradresse holen
D3D5  F0 32      BEQ $D409     soll Programm geladen werden ?
D3D7  20 25 D1   JSR $D125     nein, Dateityp feststellen
D3DA  C9 04      CMP #$04      mit Wert für relative Datei vergleichen
```

```
D3DC   90 22      BCC $D400      identisch ?
------------------------------------------------------------------
[D3CB/FFB0]
Byte aus relativer Datei holen
D3DE   20 2F D1   JSR $D12F      ja, Puffer- und Kanalnummer setzen
D3E1   B5 99      LDA $99,X      aktuellen Pufferzeiger holen und mit
D3E3   D9 44 02   CMP $0244,Y    Endposition des Puffers vergleichen
D3E6   D0 04      BNE $D3EC      Ende des gültigen Bereichs erreicht
D3E8   A9 00      LDA #$00       Pufferzeiger (Low-Byte)
D3EA   95 99      STA $99,X      wieder zurücksetzen
------------------------------------------------------------------
[CD59/D3E6]
nächstes Byte aus Datei holen
D3EC   F6 99      INC $99,X      Pufferzeiger auf nächstes Byte setzen
------------------------------------------------------------------
[CDCA/CFD5]
aktuelles Byte aus Datei holen
D3EE   A1 99      LDA ($99,X)    Byte aus Puffer lesen
D3F0   99 3E 02   STA $023E,Y    und als auszugebendes Byte merken
D3F3   B5 99      LDA $99,X      Pufferzeiger holen
D3F5   D9 44 02   CMP $0244,Y    und auf Endwert prüfen
D3F8   D0 05      BNE $D3FF      ist Ende des Datenbereichs erreicht ?
D3FA   A9 81      LDA #$81       ja, Flagwert für 'letztes Zeichen'
D3FC   99 F2 00   STA $00F2,Y    in Kanalstatustabelle übernehmen
D3FF[1] 60        RTS            zurück zur aufrufenden Routine
D400[2] 20 56 D1  JSR $D156      Zeichen aus Puffer holen
D403[1] A6 82     LDX $82        Nummer des aktuellen Kanals holen
D405   9D 3E 02   STA $023E,X    und Datenbyte zur Ausgabe bereitstellen
D408   60         RTS            zurück zur aufrufenden Routine
D409[1] AD 54 02  LDA $0254      Flag für Directory holen
D40C   F0 F2      BEQ $D400      ist Directory im Puffer ?
D40E   20 67 ED   JSR $ED67      ja, Byte aus Directory holen
D411   4C 03 D4   JMP $D403      und übergeben
------------------------------------------------------------------
[D3B8]
Fehlerkanal auslesen
D414   20 E8 D4   JSR $D4E8      aktuellen Pufferzeiger holen
D417   C9 D4      CMP #$D4       mit Wert für Fehlerpuffer vergleichen
D419   D0 18      BNE $D433      ist der Zeiger richtig gesetzt ?
D41B   A5 95      LDA $95        ja, High-Byte des Zeigers holen
D41D   C9 02      CMP #$02       und auf richtigen Wert prüfen
```

```
D41F  D0 12     BNE $D433      ist Zeiger auf Fehlerpuffer gerichtet
D421  A9 0D     LDA #$0D       ja, 'Return'
D423  85 85     STA $85        als nächstes Byte ausgeben
D425  20 23 C1  JSR $C123      Fehlerflags zurücksetzen
D428  A9 00     LDA #$00       Nummer der 'OK'-Meldung
D42A  20 C1 E6  JSR $E6C1      Meldung in Fehlerpuffer schreiben
D42D  C6 A5     DEC $A5        Zeiger auf Fehlermeldungspuffer (Low)
D42F  A9 80     LDA #$80       Flag für 'Lesen'
D431  D0 12     BNE $D445      immer Sprung nach $D445
D433[2] 20 37 D1  JSR $D137    Byte aus Fehlerpuffer holen
D436  85 85     STA $85        und als auszugebendes Byte übernehmen
D438  D0 09     BNE $D443      Ende erreicht ?
-------------------------------------------------------------------
```

[CB48]
Zeiger für Fehlermeldungspuffer setzen ($02D4)

```
D43A  A9 D4     LDA #$D4       ja, Pufferzeiger für Fehlerpuffer
D43C  20 C8 D4  JSR $D4C8      setzen
D43F  A9 02     LDA #$02       Highbyte des Zeigers
D441  95 9A     STA $9A,X      setzen
-------------------------------------------------------------------
```

[CB42/D438] Kanal für Fehlermeldung initialisieren

```
D443  A9 88     LDA #$88       Flag für 'Lesen' und 'EOI'
D445[1] 85 F7     STA $F7      für Kanal 5 setzen
D447  A5 85     LDA $85        Byte wieder holen und an
D449  8D 43 02  STA $0243      Ausgabe übergeben
D44C  60        RTS            zurück zur aufrufenden Routine
-------------------------------------------------------------------
```

[C629/C8B0/EE07]
nächsten Sektor einer Datei lesen

```
D44D  20 93 DF  JSR $DF93      Nummer des Puffers feststellen
D450  0A        ASL A          und verdoppeln
D451  AA        TAX            (da Zeigertabelle aus 2-Byte-Werten)
D452  A9 00     LDA #$00       Wert für Startposition in Puffer
D454  95 99     STA $99,X      in Pufferzeiger (Low-Byte) übernehmen
D456  A1 99     LDA ($99,X)    Spurnummer des nächsten Sektors holen
D458  F0 05     BEQ $D45F      kein weiterer Sektor vorhanden
D45A  D6 99     DEC $99,X      Pufferzeiger auf Ende setzen
D45C  4C 56 D1  JMP $D156      nächsten Sektor einlesen
D45F[1] 60        RTS          zurück zur aufrufenden Routine
-------------------------------------------------------------------
```

```
[CD39/D720/DBC9]
Jobcode für Sektor lesen übergeben
D460  A9 80     LDA #$80      Jobcode für 'Sektor lesen' setzen
D462  D0 02     BNE $D466     immer Sprung nach $D466
-----------------------------------------------------------------
[CD8C/CD9D/D790/D93A/D98A/EEAC]
Jobcode für Sektor schreiben übergeben
D464  A9 90     LDA #$90      Jobcode für 'Sektor schreiben' setzen
-----------------------------------------------------------------
[D462]
Jobcode (in A) ausführen
D466  05 7F     ORA $7F       aktuelle Laufwerksnummer in Jobcode
D468  8D 4D 02  STA $024D     setzen und Jobcode merken
D46B  A5 F9     LDA $F9       Nummer des aktuellen Puffers holen
D46D  20 D3 D6  JSR $D6D3     Spur- und Sektornummer übergeben
D470  A6 F9     LDX $F9       Nummer des aktuellen Puffers holen
D472  4C 93 D5  JMP $D593     Jobcode setzen und Job ausführen
-----------------------------------------------------------------
[C5C1/C60E/C880/CA0C/EDEB]
Sequentielle Datei zum Lesen öffnen
D475  A9 01     LDA #$01      Dateitypkennzeichen für SEQ-Datei
-----------------------------------------------------------------
[E7D5]
Datei zum Lesen öffnen
D477  8D 4A 02  STA $024A     festlegen
D47A  A9 11     LDA #$11      Nummer des internen Lesekanals (17)
D47C  85 83     STA $83       als aktuelle Sekundäradresse übernehmen
D47E  20 46 DC  JSR $DC46     Puffer belegen und Sektor einlesen
D481  A9 02     LDA #$02      Pufferzeiger auf Start des
D483  4C C8 D4  JMP $D4C8     Datenbereichs setzen
-----------------------------------------------------------------
[C9B0]
Datei zum Schreiben öffenen
D486  A9 12     LDA #$12      Nummer des internen Schreibkanals (18)
D488  85 83     STA $83       als Sekundäradresse setzen
D48A  4C DA DC  JMP $DCDA     Kanal öffnen und neuen Sektor anlegen
-----------------------------------------------------------------
[D730]
nächsten Directorysektor schreiben
D48D  20 3B DE  JSR $DE3B     aktuelle Spur- und Sektornummer holen
D490  A9 01     LDA #$01      Zahl der anzulegenden
```

```
D492  85 6F     STA $6F      Sektoren
D494  A5 69     LDA $69      normalen Sektorversatz holen
D496  48        PHA          und retten
D497  A9 03     LDA #$03     Sektorversatz für Directory auf 3
D499  85 69     STA $69      festlegen
D49B  20 2D F1  JSR $F12D    nächsten freien Sektor ermitteln
D49E  68        PLA          normalen Sektorversatz wieder
D49F  85 69     STA $69      einrichten
D4A1  A9 00     LDA #$00     Pufferzeiger auf Anfang des
D4A3  20 C8 D4  JSR $D4C8    Puffers setzen
D4A6  A5 80     LDA $80      Spurnummer des neuen Sektors in
D4A8  20 F1 CF  JSR $CFF1    aktuellen Directorysektor schreiben
D4AB  A5 81     LDA $81      Nummer des nächsten Sektors in aktuellen
D4AD  20 F1 CF  JSR $CFF1    Sektor als Verkettung übernehmen
D4B0  20 C7 D0  JSR $D0C7    aktuellen Sektor auf Diskette schreiben
D4B3  20 99 D5  JSR $D599    warten bis Jobschleife damit fertig ist
D4B6  A9 00     LDA #$00     Pufferzeiger auf Anfang
D4B8  20 C8 D4  JSR $D4C8    zurücksetzen
D4BB  20 F1 CF  JSR $CFF1    als Füllbyte in Puffer schreiben
D4BE  D0 FB     BNE $D4BB    ist gesamter Puffer gelöscht ?
D4C0  20 F1 CF  JSR $CFF1    ja, Kennzeichen für letzten Sektor
D4C3  A9 FF     LDA #$FF     Zahl der gültigen Bytes des Sektors
D4C5  4C F1 CF  JMP $CFF1    in Sektor schreiben

[C614/C896/CA14/CA32/CAB4/CD47/CD83/D16F/D1AB/D1C3/D43C/D483/D4A3/D4B8]
[D740/D914/DA42/DB92/DCA0/DD6F/DE97/DFFA/E04F/E27A/E476/E4A3/E4C0/E4DB]
[ECA9/EDF0]
Pufferzeiger auf angegebene Position setzen
D4C8  85 6F     STA $6F      neue Position merken
D4CA  20 93 DF  JSR $DF93    Nummer des aktuellen Puffers holen
D4CD  0A        ASL A        und verdoppeln (da Zeigertabelle aus
D4CE  AA        TAX          2-Byte-Zeigern besteht)
D4CF  B5 9A     LDA $9A,X    Pufferzeiger (High-Byte)
D4D1  85 95     STA $95      holen und setzen
D4D3  A5 6F     LDA $6F      Low-Byte des Pufferzeigers holen
D4D5  95 99     STA $99,X    merken und als aktuellen
D4D7  85 94     STA $94      Pufferzeiger setzen
D4D9  60        RTS          zurück zur aufrufenden Routine
```

```
[C1BA/DAD1/E653]
Interne Kanäle schließen
D4DA  A9 11      LDA #$11      Nummer des internen Lesekanals (17)
D4DC  85 83      STA $83       als aktuelle Sekundäradresse setzen
D4DE  20 27 D2   JSR $D227     Kanal schließen
D4E1  A9 12      LDA #$12      Nummer des internen Schreibkanals (18)
D4E3  85 83      STA $83       als aktuelle Sekundäradresse speichern
D4E5  4C 27 D2   JMP $D227     und Kanal schließen
--------------------------------------------------------------------
[C5D7/C6E5/CD76/CFDC/D1C8/D414/DB6A/DB76/DFEA/E182/E1A9]
aktuellen Pufferzeiger festlegen
D4E8  20 93 DF   JSR $DF93     Nummer des aktuellen Puffers holen
--------------------------------------------------------------------
[DF49]
Pufferzeiger setzen (Puffernummer in A)
D4EB  0A         ASL A         verdoppeln (da Zeigertabelle aus
D4EC  AA         TAX           2-Byte-Werten besteht)
D4ED  B5 9A      LDA $9A,X     Zeiger auf Position in Puffer holen
D4EF  85 95      STA $95       und als aktuellen
D4F1  B5 99      LDA $99,X     Pufferzeiger
D4F3  85 94      STA $94       übernehmen
D4F5  60         RTS           zurück zur aufrufenden Routine
--------------------------------------------------------------------
[C5D1/CF39/CF40/E00E/E39F]
beliebiges Byte aus Puffer lesen (A muß Position des Zeichen enthalten)
D4F6  85 71      STA $71       Pufferposition merken
D4F8  20 93 DF   JSR $DF93     aktuelle Puffernummer feststellen
D4FB  AA         TAX           und merken
D4FC  BD E0 FE   LDA $FEE0,X   High-Byte der zugehörigen Pufferadresse
D4FF  85 72      STA $72       holen und setzen
D501  A0 00      LDY #$00      Pufferzeiger initialisieren
D503  B1 71      LDA ($71),Y   und Byte aus Pufferposition holen
D505  60         RTS           zurück zur aufrufenden Routine
--------------------------------------------------------------------
[D0D0/DE32]
Spur- und Sektornummer auf Gültigkeit überprüfen, dann Jobcode setzen
D506  BD 5B 02   LDA $025B,X   dem Puffer zugeordneter Jobcode holen
D509  29 01      AND #$01      und Laufwerksnummer daraus feststellen
D50B  0D 4D 02   ORA $024D     aktuellen Jobcode einblenden
D50E  48         PHA           und merken
D50F  86 F9      STX $F9       Puffernummer festhalten
```

```
D511    8A          TXA          und Nummer
D512    0A          ASL A        verdoppeln (da nächste Tabelle aus
D513    AA          TAX          2-Byte-Werten besteht)
D514    B5 07       LDA $07,X    Sektornummer des Jobs holen
D516    8D 4D 02    STA $024D    und merken
D519    B5 06       LDA $06,X    Nummer der Spur für den Job feststellen
D51B    F0 2D       BEQ $D54A    ist keine Spur angewählt (0) ?
D51D    CD AC 02    CMP $02AC    nein, und auf größte Spur +1 testen
D520    B0 28       BCS $D54A    ist die Spurnummer im erlaubten Bereich?
D522    AA          TAX          ja, Spurnummer merken
D523    68          PLA          und Jobcode wieder zurückholen
D524    48          PHA          und sofort wieder merken
D525    29 F0       AND #$F0     reinen Jobcode isolieren
D527    C9 90       CMP #$90     und mit Code für 'Schreiben' vergleichen
D529    D0 4F       BNE $D57A    identisch ?
D52B    68          PLA          ja, ganzen Jobcode wieder holen
D52C    48          PHA          und sofort wieder merken
D52D    4A          LSR A        Bitflag für Laufwerknummer ins Carry
D52E    B0 05       BCS $D535    ist Laufwerk 1 angewählt ?
D530    AD 01 01    LDA $0101    nein, Formatkennzeichen von Laufwerk 0
D533    90 03       BCC $D538    immer Sprung nach $D538
D535[1] AD 02 01    LDA $0102    Formatkennzeichen von Laufwerk 1 holen
D538[1] F0 05       BEQ $D53F    immer Sprung nach $D53F
D53A    CD D5 FE    CMP $FED5    mit Kennzeichen 'A' vergleichen
D53D    D0 33       BNE $D572    ist richtiges Format erkannt ?
D53F[1] 8A          TXA          ja, Nummer der Spur wieder holen
D540    20 4B F2    JSR $F24B    dazugehörige größte Sektornummer holen
D543    CD 4D 02    CMP $024D    mit Sektornummer des Jobs vergleichen
D546    F0 02       BEQ $D54A    ist maximale Zahl erreicht ?
D548    B0 30       BCS $D57A    nein, ist Sektornummer erlaubt ?
D54A[3] 20 52 D5    JSR $D552    nein, Spur- und Sektornummer des Jobs
D54D[4] A9 66       LDA #$66     nochmal holen und Fehlermeldung
D54F    4C 45 E6    JMP $E645    '66 Illegal Track or Sector' ausgeben
---------------------------------------------------------------------------
[D54A/D572]
Spur- und Sektornummer des aktuellen Jobs aus Jobspeicher holen
D552    A5 F9       LDA $F9      Nummer des aktuellen Jobs (Puffers)
D554    0A          ASL A        holen und verdoppeln
D555    AA          TAX          (da Tabelle aus 2-Byte-Werten besteht)
D556    B5 06       LDA $06,X    Spurnummer des Jobs aus Tabelle holen
D558    85 80       STA $80      und  als aktuelle Spur merken
```

```
D55A  B5 07     LDA $07,X     Sektornummer des Jobs feststellen und
D55C  85 81     STA $81       als aktuelle Sektornummer speichern
D55E  60        RTS           zurück zur aufrufenden Routine
-----------------------------------------------------------------------
[CE08/EDE5]
aktuelle Spur- und Sektornummer auf erlaubten Bereich überprüfen
D55F  A5 80     LDA $80       aktuelle Sprunummer holen
D561  F0 EA     BEQ $D54D     keine Spur gesetzt ?
D563  CD AC 02  CMP $02AC     nein, auf max. zulässiger Spur prüfen
D566  B0 E5     BCS $D54D     ist die Spurnummer erlaubt (kleiner) ?
D568  20 4B F2  JSR $F24B     ja, Zahl der Sektoren der Spur holen und
D56B  C5 81     CMP $81       mit aktueller Sektornummer vergleichen
D56D  F0 DE     BEQ $D54D     ist die Sektornummer eins zu groß ?
D56F  90 DC     BCC $D54D     nein, ist die Nummer noch größer ?
D571  60        RTS           nein, zurück zur aufrufenden Routine
-----------------------------------------------------------------------
[D53D/EE53]
Fehlermeldung bei falschem Formatkennzeichen ausgeben
D572  20 52 D5  JSR $D552     Spur- und Sektornummer des Jobs holen
D575  A9 73     LDA #$73      und Fehlermeldung
D577  4C 45 E6  JMP $E645     '73 CBM DOS V3.0 1571' ausgeben
-----------------------------------------------------------------------
[D529/D548]
Job für aktuellen Puffer an Jobschleife übergeben
(Routine kann nicht mit 'JSR' angesprungen werden, da Stack den Jobcode
enthalten muß und nicht die Rücksprungadresse)
D57A  A6 F9     LDX $F9       Nummer des aktuellen Puffers holen
D57C  68        PLA           zu setzender Jobcode holen
D57D  8D 4D 02  STA $024D     und als aktuellen Jobcode speichern
D580  95 00     STA $00,X     an Jobschleife übergeben
D582  9D 5B 02  STA $025B,X   dem aktuellen Puffer zuordnen
D585  60        RTS           zurück zur aufrufenden Routine
-----------------------------------------------------------------------
[A5D1/A66E/A693/A6BA]
Jobcode für Lesen an Jobschleife übergeben und warten bis ausgeführt
D586  A9 80     LDA #$80      Jobcode für 'Sektor lesen'
D588  D0 02     BNE $D58C     immer Sprung nach $D58C
-----------------------------------------------------------------------
[A594/A5A4/A5C5]
Jobcode für Schreiben an Jobschleife übergeben und warten bis ausgeführt
D58A  A9 90     LDA #$90      Jobcode für 'Sektor schreiben'
```

```
--------------------------------------------------------------------------
[A6E5/A70E/D588]
Job für aktuelles Laufwerk ausführen (Jobcode in A)
D58C  05 7F      ORA $7F         aktuelles Laufwerk in Jobcode übernehmen
D58E  A6 F9      LDX $F9         Nummer des zuständigen uffers holen
--------------------------------------------------------------------------
[DC3D]
Jobcode ausführen (Jobcode in A, Puffernummer in X)
D590  8D 4D 02   STA $024D       und aktuellen Jobcode merken
--------------------------------------------------------------------------
[D472/DF42]  Job ausführen
D593  AD 4D 02   LDA $024D       Jobcode holen, Spur- und Sektor-
D596  20 0E D5   JSR $D50E       parameter überprüfen und an Jobschleife
--------------------------------------------------------------------------
[869A/C8BE/CAAC/CAC6/D0A1/D4B3/DB9F/DC95/DD6A/DD84/DDF9/E068/E430/E4A9]
[E4F0/CF73/E05A]
warten, bis Job ausgeführt ist und Fehlermeldung bereitstellen
D599  20 A6 D5   JSR $D5A6       Jobausführung kontrollieren
D59C  B0 FB      BCS $D599       ist Job schon zu Ende ?
D59E  48         PHA             ja, Rückmeldung des Jobs merken
D59F  A9 00      LDA #$00        Flag für 'Fehler bei Job aufgetreten'
D5A1  8D 98 02   STA $0298       löschen
D5A4  68         PLA             und Rückmeldung wieder holen
D5A5  60         RTS             zurück zur aufrufenden Routine
--------------------------------------------------------------------------
[D599]
Ausführung des aktuellen Jobs überwachen
D5A6  B5 00      LDA $00,X       Jobcode aus Jobspeicher holen
D5A8  30 1A      BMI $D5C4       ist Job noch in Bearbeitung ?
D5AA  C9 02      CMP #$02        nein, Rückmeldung auf 'Ok' testen
D5AC  90 14      BCC $D5C2       ist der Job ordungsgemäß ausgeführt ?
D5AE  C9 08      CMP #$08        nein, mit 'Write Protect On' vergleichen
D5B0  F0 08      BEQ $D5BA       ist die Schreibschutzkerbe zugeklebt ?
D5B2  C9 0B      CMP #$0B        nein, mit 'Disk ID Mismatch' vergleichen
D5B4  F0 04      BEQ $D5BA       wurde eine falsche ID gefunden ?
D5B6  C9 0F      CMP #$0F        nein, mit 'Drive Not Ready' vergleichen
D5B8  D0 0C      BNE $D5C6       keine formatierte Diskette eingelegt ?
D5BA2 2C 98 02   BIT $0298       ja, Fehlerflag prüfen
D5BD  30 03      BMI $D5C2       wurde schon ein Fehler gemeldet ?
D5BF  4C 3F D6   JMP $D63F       nein, Fehlermeldung ausgeben
D5C22 18         CLC             Flag für 'Job beendet' setzen
```

```
D5C3  60         RTS          zurück zur aufrufenden Routine
D5C4  38         SEC          Flag für 'Job noch nicht beendet' setzen
D5C5  60         RTS          zurück zur aufrufenden Routine
-----------------------------------------------------------------------
[D5B8/D644:A6CE]
Kopf bei aufgetretenem Lesefehler neben Spur setzen und nochmal versuchen
D5C6  98         TYA          Y-Register retten
D5C7  48         PHA          (da es von der Routine verändert wird)
D5C8  A5 7F      LDA $7F      aktuelle Laufwerksnummer holen
D5CA  48         PHA          und merken
D5CB  BD 5B 02   LDA $025B,X  zum Puffer gehörenden Jobcode holen
D5CE  29 01      AND #$01     und angesprochenes Laufwerk feststellen
D5D0  85 7F      STA $7F      Nummer als aktuelles Laufwerk speichern
D5D2  A8         TAY          und die zum Laufwerk gehörende Bitmaske
D5D3  B9 CA FE   LDA $FECA,Y  holen, um LED am Laufwerk anzuschalten
D5D6  8D 6D 02   STA $026D    Maske für LED-Blinken merken
D5D9  20 A6 D6   JSR $D6A6    ($6A) Leseversuche durchführen
D5DC  C9 02      CMP #$02     Rückmeldung mit 'Ok' vergleichen
D5DE  B0 03      BCS $D5E3    wurde letzter Job fehlerfrei ausgeführt
D5E0  4C 6D D6   JMP $D66D    ja, Routine beenden
D5E3  BD 5B 02   LDA $025B,X  aktuellen Jobcode holen
D5E6  29 F0      AND #$F0     Befehlbits isolieren
D5E8  48         PHA          und merken
D5E9  C9 90      CMP #$90     mit Wert für 'Schreiben' vergleichen
D5EB  D0 07      BNE $D5F4    wurde ein Sektor geschrieben ?
D5ED  A5 7F      LDA $7F      ja, Nummer des Laufwerks holen
D5EF  09 B8      ORA #$B8     und Jobcode für 'Sektor suchen' setzen
D5F1  9D 5B 02   STA $025B,X  Jobcode dem aktuellen Puffer zuordnen
D5F4  24 6A      BIT $6A      Flag für 'nicht neben Spur suchen'
D5F6  70 39      BVS $D631    ist Flag gesetzt ?
D5F8  A9 00      LDA #$00     nein, Zeiger initialisieren :
D5FA  8D 99 02   STA $0299    Zeiger auf Position neben Spur
D5FD  8D 9A 02   STA $029A    Zeiger auf Phase der Suche neben Spur
D600  AC 99 02   LDY $0299    Positionierphase feststellen
D603  AD 9A 02   LDA $029A    aktuelles Steuerbyte für Kopfbewegung
D606  38         SEC          holen und Wert für einen Rückkehr auf
D607  F9 DB FE   SBC $FEDB,Y  Ausgangsposition bei der
D60A  8D 9A 02   STA $029A    Positionierung neben der Spur ermitteln
D60D  B9 DB FE   LDA $FEDB,Y  Steuerbyte für Halbschritt neben Spur
D610  20 A1 FF   JSR $FFA1    holen und Kopfbewegung ausführen
D613  EE 99 02   INC $0299    Zähler auf nächstes Steuerbyte setzen
```

```
D616     20 A6 D6   JSR $D6A6     ($6A) Leseversuche ausführen
D619     C9 02      CMP #$02      Rückmeldung auf Wert für 'Ok' prüfen
D61B     90 08      BCC $D625     ist ein Fehler aufgetreten ?
D61D     AC 99 02   LDY $0299     ja, Zähler für Positionierphase holen
D620     B9 DB FE   LDA $FEDB,Y   nächstes Positionierkommando holen
D623     D0 DB      BNE $D600     Ende der Suchfolge ?
D625[1]  AD 9A 02   LDA $029A     ja, Steuerwert zur Rückkehr auf Spur
D628     20 A6 FF   JSR $FFA6     holen und Lesen nochmal versuchen
D62B     B5 00      LDA $00,X     Rückmeldung der Jobschleife holen
D62D     C9 02      CMP #$02      und mit Wert für 'Ok' vergleichen
D62F     90 2B      BCC $D65C     war Leseversuch erfolgreich ?
D631[1]  24 6A      BIT $6A       nein, Flag für 'Kopf auf Spur 0' prüfen
D633     10 0F      BPL $D644     soll Kopf neu justiert werden (Bump) ?
D635[1]  68         PLA           nein, Befehlscode wieder holen und auf
D636     C9 90      CMP #$90      Job für 'Sektor schreiben' prüfen
D638     D0 05      BNE $D63F     identisch ?
D63A     05 7F      ORA $7F       ja, Nummer des Laufwerks setzen und
D63C     9D 5B 02   STA $025B,X   als Jobcode aktuellem Puffer zuordnen
D63F[3]  B5 00      LDA $00,X     Rückmeldung des Jobs holen
D641     20 0A E6   JSR $E60A     und Fehlermeldung bereitstellen
D644[2]  68         PLA           reinen Befehlscode wieder holen
D645     2C 98 02   BIT $0298     Fehlerflag testen
D648     30 23      BMI $D66D     ist schon ein Fehler aufgetreten
D64A     48         PHA           nein, Jobcode merken
D64B     A9 C0      LDA #$C0      und Jobcode für 'Kopf neu justieren'
D64D     05 7F      ORA $7F       (Bump) für aktuelles Laufwerk
D64F     95 00      STA $00,X     setzen
D651     20 B6 9F   JSR $9FB6     Jobschleife starten und Job ausführen
D654     EA         NOP           [durch Modifizierung des 1541 ROM]
D655     20 A6 D6   JSR $D6A6     Job noch ($6A) mal ausführen
D658     C9 02      CMP #$02      Rückmeldung mit 'Ok' vergleichen
D65A     B0 D9      BCS $D635     wurde letzter erfolgreich ausgeführt ?
D65C[1]  68         PLA           ja, Jobcode wieder holen
D65D     C9 90      CMP #$90      und mit Wert für 'Schreiben' vergleichen
D65F     D0 0C      BNE $D66D     sollte Sektor geschrieben werden ?
D661     05 7F      ORA $7F       ja, Laufwerksnummer setzen
D663     9D 5B 02   STA $025B,X   und Jobcode aktuellem Puffer zuordnen
D666     20 A6 D6   JSR $D6A6     ($6A) mal Sektor schreiben versuchen
D669     C9 02      CMP #$02      Rückmeldung mit 'Ok' vergleichen
D66B     B0 D2      BCS $D63F     war Schreiben erfolgreich ?
D66D[3]  68         PLA           ja, Nummer des aktuellen Laufwerks
```

```
D66E   85 7F      STA $7F       wieder herstellen
D670   68         PLA           Y-Register wieder
D671   A8         TAY           zurücksetzen
D672   B5 00      LDA $00,X     Rückmeldung der Jobschleife holen
D674   18         CLC           Flag für 'Job beendet' setzen
D675   60         RTS           zurück zur aufrufenden Routine
-----------------------------------------------------------------------
[FF99/FF9C]
Kopf um die im Akku angegebenen Halbspurschritte bewegen
(Bit7 =1 Schritte nach innen; Bit7 =0 Schritte nach außen)
D676   C9 00      CMP #$00      Inhalt des Akkus überprüfen
D678   F0 18      BEQ $D692     ist ein Wert der Schritte angegeben ?
D67A   30 0C      BMI $D688     ja, soll Kopf nach außen bewegt werden ?
D67C[1] A0 01     LDY #$01      Wert für Halbspurschritt nach innen
D67E   20 93 D6   JSR $D693     Kopf neu setzen
D681   38         SEC           und Zähler für Zahl
D682   E9 01      SBC #$01      der Halbspurschritte erniedrigen
D684   D0 F6      BNE $D67C     schon alle Schritte gemacht ?
D686   F0 0A      BEQ $D692     ja, immer Sprung nach $D692
D688[2] A0 FF     LDY #$FF      Wert für Halbspurschritt nach außen
D68A   20 93 D6   JSR $D693     Kopf neu setzen
D68D   18         CLC           und Zähler für Zahl
D68E   69 01      ADC #$01      der Halbspurschritte erhöhen
D690   D0 F6      BNE $D688     schon alle Schritte gemacht ?
D692[2] 60        RTS           ja, zurück zur aufrufenden Routine
-----------------------------------------------------------------------
[D67E/D68A]
Werte um Kopf zu bewegen an Jobschleife übergeben
D693   48         PHA           Akku retten
D694   98         TYA           Wert für Kopfpositionierung holen
D695   A4 7F      LDY $7F       Nummer des aktuellen Laufwerks holen
D697   99 FE 02   STA $02FE,Y   und Steuerbyte an Jobschleife übergeben
D69A[1] D9 FE 02  CMP $02FE,Y   Wert wieder holen
D69D   F0 FB      BEQ $D69A     wurde Wert angenommen und Kopf gesetzt ?
D69F   A9 00      LDA #$00      ja, Jobregister
D6A1   99 FE 02   STA $02FE,Y   löschen
D6A4   68         PLA           Akku wieder herstellen
D6A5   60         RTS           zurück zur aufrufenden Routine
-----------------------------------------------------------------------
```

[D5D9/D616/D655/D666]

Jobcode solange ausführen bis erfolgreich oder Zähler in $6A auf Null

```
D6A6      A5 6A      LDA $6A      Zahl der Versuche holen
D6A8      29 3F      AND #$3F     und auf Bereich von 0 bis 63 begrenzen
D6AA      A8         TAY          Zähler setzen
D6AB [1]  AD 6D 02   LDA $026D    Maske für LED einschalten
D6AE      4D 00 1C   EOR $1C00    Bit für LED im Laufwerkssteuerregister
D6B1      8D 00 1C   STA $1C00    umschalten (LED flackert)
D6B4      BD 5B 02   LDA $025B,X  Jobcode des aktuellen Puffers holen
D6B7      95 00      STA $00,X    und an Jobschleife übergeben
D6B9      20 B6 9E   JSR $9EB6    Jobschleife starten und Job ausführen
D6BC      EA         NOP          [durch Modifizierung des 1541 ROM]
D6BD      C9 02      CMP #$02     Rückmeldung mit 'Ok' vergleichen
D6BF      90 03      BCC $D6C4    wurde Job erfolgreich ausgeführt ?
D6C1      88         DEY          nein, Zahl der Versuche erniedrigen
D6C2      D0 E7      BNE $D6AB    noch weitere Versuche durchführen ?
D6C4 [1]  48         PHA          nein, Jobmeldung merken
D6C5      AD 6D 02   LDA $026D    Maske für 'LED ein' holen
D6C8      0D 00 1C   ORA $1C00    und restliche Bits des Steuerregisters
D6CB      8D 00 1C   STA $1C00    einblenden und Register setzen
D6CE      68         PLA          Rückmeldung der Jobschleife wieder holen
D6CF      60         RTS          zurück zur aufrufenden Routine
```

[D09B/D0BA/D186/DCE2/DE7F/E3B9/E3CB]

aktuelle Spur- und Sektornummer an Jobschleife übergeben

```
D6D0      20 93 DF   JSR $DF93    Nummer des aktuellen Puffers holen
```

[A414/C8D7/D03A/D46D/DC8F/DD2E/DF3D]

Spur- und Sektor an Jobschleife übergeben (Puffer in A)

```
D6D3      0A         ASL A        und verdoppeln (da Jobtabelle aus
D6D4      A8         TAY          2-Byte-Werten besteht)
D6D5      A5 80      LDA $80      Nummer der aktuellen Spur
D6D7      99 06 00   STA $0006,Y  an Jobschleife übergeben
D6DA      A5 81      LDA $81      Nummer des aktuellen Sektors
D6DC      99 07 00   STA $0007,Y  für Jobschleife speichern
D6DF      A5 7F      LDA $7F      aktuelle Laufwerksnummer holen
D6E1      0A         ASL A        verdoppeln
D6E2      AA         TAX          und merken
D6E3      60         RTS          zurück zur aufrufenden Routine
```

[C9B3/D9EC]

Dateieintrag im Directory abschließen

```
D6E4  A5 83     LDA $83       aktuelle Sekundäradresse holen und
D6E6  48        PHA           retten
D6E7  A5 82     LDA $82       aktuelle Kanalnummer holen und
D6E9  48        PHA           retten
D6EA  A5 81     LDA $81       aktuelle Sektornummer holen und
D6EC  48        PHA           retten
D6ED  A5 80     LDA $80       aktuelle Spurnummer holen und
D6EF  48        PHA           retten
D6F0  A9 11     LDA #$11      Nummer des internen Lesekanals (17)
D6F2  85 83     STA $83       als aktuelle Kanalnummer setzen
D6F4  20 3B DE  JSR $DE3B     Spur- und Sektornummer festlegen
D6F7  AD 4A 02  LDA $024A     aktueller Dateityp
D6FA  48        PHA           holen und retten
D6FB  A5 E2     LDA $E2       Laufwerksnummer der neuen Datei
D6FD  29 01     AND #$01      herstellen und
D6FF  85 7F     STA $7F       als aktuelle Laufwerksnummer festlegen
D701  A6 F9     LDX $F9       Nummer des aktuellen Puffers
D703  5D 5B 02  EOR $025B,X   Laufwerksnummer des zugehörigen Jobcodes
D706  4A        LSR A         holen und mit Aktueller vergleichen
D707  90 0C     BCC $D715     ist angesprochendes Laufwerk identisch ?
D709  A2 01     LDX #$01      Zeiger auf gültigen
D70B  8E 92 02  STX $0292     Eintrag setzen
D70E  20 AC C5  JSR $C5AC     nächsten freien Eintrag suchen
D711  F0 1D     BEQ $D730     ist alles belegt ?
D713  D0 28     BNE $D73D     nein, Eintrag in Directory schreiben
D715  AD 91 02  LDA $0291     Nummer des Directorysektors holen
D718  F0 0C     BEQ $D726     ist Sektornummer gesetzt ?
D71A  C5 81     CMP $81       ja, mit aktueller Sektornr. vergleichen
D71C  F0 1F     BEQ $D73D     identisch ?
D71E  85 81     STA $81       nein, Nummer des aktuellen Sektors holen
D720  20 60 D4  JSR $D460     Sektor in Puffer einlesen
D723  4C 3D D7  JMP $D73D     neuen Eintrag anlegen
D726  A9 01     LDA #$01      Zeiger auf gültigen
D728  8D 92 02  STA $0292     Dateieintrag setzen
D72B  20 17 C6  JSR $C617     letzten Sektor des Directory holen
D72E  D0 0D     BNE $D73D     noch ein Eintrag frei ?
D730  20 8D D4  JSR $D48D     nein, neuen Directorysektor anlegen
D733  A5 81     LDA $81       Nummer des Sektors holen
D735  8D 91 02  STA $0291     und in Zeiger für Direktorysektor setzen
```

```
D738     A9 02      LDA #$02       Pufferzeiger auf Beginn des
D73A     8D 92 02   STA $0292      Datenbereichs initialisieren
D73D[4]  AD 92 02   LDA $0292      aktuelle Position des Zeigers
D740     20 C8 D4   JSR $D4C8      Pufferzeiger setzen
D743     68         PLA            aktueller Dateityp zurückholen
D744     8D 4A 02   STA $024A      und wieder setzen
D747     C9 04      CMP #$04       mit Wert für Relative Datei vegleichen
D749     D0 02      BNE $D74D      liegt eine relative Datei vor ?
D74B     09 80      ORA #$80       ja, Datei als geschlossen kennzeichnen
D74D[1]  20 F1 CF   JSR $CFF1      Dateityp in Directory eintragen
D750     68         PLA            Spurnummer des ersten Dateisektors
D751     8D 80 02   STA $0280      wieder holen und speichern
D754     20 F1 CF   JSR $CFF1      Spurnummer in Directory schreiben
D757     68         PLA            Nummer des ersten Sektors der Datei
D758     8D 85 02   STA $0285      holen und merken
D75B     20 F1 CF   JSR $CFF1      Sektornummer in Directory schreiben
D75E     20 93 DF   JSR $DF93      Nummer des Directorypuffers holen
D761     A8         TAY            und merken
D762     AD 7A 02   LDA $027A      Position des Dateinamens im
D765     AA         TAX            Eingabepuffer holen und merken
D766     A9 10      LDA #$10       Länge des Dateinamens
D768     20 6E C6   JSR $C66E      Dateinamen in Directory schreiben
D76B     A0 10      LDY #$10       Pufferzeiger auf Ende des Dateinamens
D76D     A9 00      LDA #$00       Leerbytes zum Auffüllen des Eintrags
D76F[1]  91 94      STA ($94),Y    in Puffer schreiben
D771     C8         INY            Pufferzeiger auf nächstes Byte
D772     C0 1B      CPY #$1B       und mit Endwert vergleichen
D774     90 F9      BCC $D76F      schon ganzer Puffer gefüllt ?
D776     AD 4A 02   LDA $024A      ja, aktuellen Dateityp holen
D779     C9 04      CMP #$04       mit Wert für Relative Datei vergleichen
D77B     D0 13      BNE $D790      ist eine Relative Datei einzutragen ?
D77D     A0 10      LDY #$10       ja, Pufferzeiger auf Ende des Namens
D77F     AD 59 02   LDA $0259      Spurnummer des ersten Side-Sektors
D782     91 94      STA ($94),Y    holen und in Eintrag schreiben
D784     C8         INY            Pufferzeiger auf nächste Position
D785     AD 5A 02   LDA $025A      Sektornummer holen und in Directory-
D788     91 94      STA ($94),Y    puffer schreiben
D78A     C8         INY            Pufferzeiger auf nächstes Byte
D78B     AD 58 02   LDA $0258      Länge des Records holen und
D78E     91 94      STA ($94),Y    in Directory eintragen
D790[1]  20 64 D4   JSR $D464      Directorysektor auf Diskette schreiben
```

```
D793  68        PLA           Nummer des aktuellen Kanals holen
D794  85 82     STA $82       und wieder setzen
D796  AA        TAX           Kanalnummer merken
D797  68        PLA           aktuelle Sekundäradresse wieder
D798  85 83     STA $83       zurückholen und setzen
D79A  AD 91 02  LDA $0291     Spurnummer des Dateieintrags holen
D79D  85 D8     STA $D8       und merken
D79F  9D 60 02  STA $0260,X   und Puffer zuordnen
D7A2  AD 92 02  LDA $0292     Sektornummer des Dateieintrags
D7A5  85 DD     STA $DD       holen und merken
D7A7  9D 66 02  STA $0266,X   Nummer dem Directorypuffer zuordnen
D7AA  AD 4A 02  LDA $024A     Typ der Datei holen
D7AD  85 E7     STA $E7       und merken
D7AF  A5 7F     LDA $7F       Nummer des aktuellen Laufwerks holen
D7B1  85 E2     STA $E2       und Dateieintrag zuordnen
D7B3  60        RTS           zurück zur aufrufenden Routine
-----------------------------------------------------------------------
[BF5D/C15D]
Open-Befehl mit Sekundäradresse 0 bis 14 annehmen
D7B4  A5 83     LDA $83       aktuelle Sekundäradresse holen
D7B6  8D 4C 02  STA $024C     und merken
D7B9  20 B3 C2  JSR $C2B3     Zeiger für Befehlsstring setzen
D7BC  8E 2A 02  STX $022A     Kanalnummer des Befehls löschen (0)
D7BF  AE 00 02  LDX $0200     erstes Zeichen des Eingabepuffers holen
D7C2  AD 4C 02  LDA $024C     Sekundäradresse holen
D7C5  D0 2C     BNE $D7F3     liegt ein Load-Befehl vor ?
D7C7  E0 2A     CPX #$2A      ja, erstes Zeichen auf '*' Joker prüfen
D7C9  D0 28     BNE $D7F3     ersten Diretoryeintrag laden ?
D7CB  A5 7E     LDA $7E       ja, letzte Spurnummer holen
D7CD  F0 4D     BEQ $D81C     ist Nummer gesetzt ?
D7CF  85 80     STA $80       ja, dann als aktuelle Spur übernehmen
D7D1  AD 6E 02  LDA $026E     Nummer des letzten aktiven Laufwerks
D7D4  85 7F     STA $7F       holen und als aktuelles Laufwerk setzen
D7D6  85 E2     STA $E2       Laufwerk der Datei zuordnen
D7D8  A9 02     LDA #$02      Flag für Joker
D7DA  85 E7     STA $E7       setzen
D7DC  AD 6F 02  LDA $026F     letzten bearbeiteten Sektor holen und
D7DF  85 81     STA $81       als aktuelle Sektornummer übernehmen
D7E1  20 00 C1  JSR $C100     LED am aktuellen Laufwerk einschalten
D7E4  20 46 DC  JSR $DC46     Puffer belegen und Sektor einlesen
D7E7  A9 04     LDA #$04      Bitflag für Programmdatei
```

```
D7E9     05 7F      ORA $7F       aktuelles Laufwerk mit einblenden
D7EB[1]  A6 82      LDX $82       Nummer des aktuellen Kanals holen
D7ED     99 EC 00   STA $00EC,Y   und Dateitypflag in dem Kanal zuordnen
D7F0     4C 94 C1   JMP $C194     'Ok' Meldung bereitstellen
D7F3[2]  E0 24      CPX #$24      erstes Zeichen mit '$' vergleichen
D7F5     D0 1E      BNE $D815     soll Directory geladen werden ?
D7F7     AD 4C 02   LDA $024C     ja, Sekundäradresse wieder holen
D7FA     D0 03      BNE $D7FF     Directory als Programm laden ?
D7FC     4C 55 DA   JMP $DA55     ja, Directory in Basic-Programm wandeln
D7FF[1]  20 D1 C1   JSR $C1D1     Zeiger für Parameter im Befehl setzen
D802     AD 85 FE   LDA $FE85     Nummer der Directoryspur
D805     85 80      STA $80       als aktuelle Spur merken
D807     A9 00      LDA #$00      Startsektor des Directory als
D809     85 81      STA $81       aktuelle Spurnummer setzen
D80B     20 46 DC   JSR $DC46     Puffer belegen und Sektor lesen
D80E     A5 7F      LDA $7F       Nummer des aktuellen Laufwerks holen
D810     09 02      ORA #$02      Flag fur SEQ-Datei setzen und Directory
D812     4C EB D7   JMP $D7EB     als Datei kennzeichnen und Ende
D815[1]  E0 23      CPX #$23      erstes Zeichen mit '#' vergleichen
D817     D0 12      BNE $D82B     Direktzugriffskanal öffnen ?
D819     4C 84 CB   JMP $CB84     ja, Direktzugriffsdatei öffnen
D81C[1]  A9 02      LDA #$02      Kennzeichen für Programm-Datei
D81E     8D 96 02   STA $0296     setzen
D821     A9 00      LDA #$00      Laufwerk 0 als aktuelles
D823     85 7F      STA $7F       Laufwerk festlegen
D825     8D 8E 02   STA $028E     Zeiger auf letztes Laufwerk setzen
D828     20 42 D0   JSR $D042     BAM in Puffer lesen
D82B[1]  20 E5 C1   JSR $C1E5     Befehlsstring nach ':' durchsuchen
D82E     D0 04      BNE $D834     gefunden ?
D830     A2 00      LDX #$00      ja, Startposition des Parameters
D832     F0 0C      BEQ $D840     immer Sprung nach $D840
D834[1]  8A         TXA           Zahl der Parameter holen
D835     F0 05      BEQ $D83C     durch Komma getrennte Parameter ?
D837     A9 30      LDA #$30      ja, Fehlermeldung
D839     4C C8 C1   JMP $C1C8     '30 Syntax Error' ausgeben
D83C[1]  88         DEY           Zeiger auf Position des ':' setzen
D83D     F0 01      BEQ $D840     Beginn des Parameters erreicht ?
D83F     88         DEY           Zeiger auf Laufwerksangabe setzen
D840[2]  8C 7A 02   STY $027A     und Position merken
D843     A9 8D      LDA #$8D      Kennzeichen für Befehlsstringende
D845     20 68 C2   JSR $C268     im Eingabepuffer suchen
```

```
D848    E8          INX          Zahl der gefundenen Parameter
D849    8E 78 02    STX $0278    die durch Komma getrennt sind merken
D84C    20 12 C3    JSR $C312    Laufwerksnummer holen und setzen
D84F    20 CA C3    JSR $C3CA    Nummer des Laufwerks überprüfen
D852    20 9D C4    JSR $C49D    Dateieintrag im Directory suchen
D855    A2 00       LDX #$00     Zeiger löschen :
D857    8E 58 02    STX $0258    Länge eines Records
D85A    8E 97 02    STX $0297    Dateibetriebsart
D85D    8E 4A 02    STX $024A    Dateityp
D860    E8          INX          nächsten Dateinamen anwählen
D861    EC 77 02    CPX $0277    auf Zahl der vorhandenen Namen prüfen
D864    B0 10       BCS $D876    sind noch weitere Angaben vorhanden ?
D866    20 09 DA    JSR $DA09    ja, Dateityp und Betriebsart holen
D869    E8          INX          Zeiger auf nächsten Dateinamen
D86A    EC 77 02    CPX $0277    auf Zahl der vorhandenen Namen prüfen
D86D    B0 07       BCS $D876    sind alle Namen bearbeitet ?
D86F    C0 04       CPY #$04     nein, Dateityp auf 'REL' prüfen
D871    F0 3E       BEQ $D8B1    liegt relative Datei vor ?
D873    20 09 DA    JSR $DA09    Dateityp und Betriebsart holen
D876[4] AE 4C 02    LDX $024C    Sekundäradresse wieder holen
D879    86 83       STX $83      und setzen; mit Wert für
D87B    E0 02       CPX #$02     Beginn der Dateikanäle vergleichen
D87D    B0 12       BCS $D891    ist Kanalnummer größer 2 ?
D87F    8E 97 02    STX $0297    nein, Flag f. Lesen o. Schreiben setzen
D882    A9 40       LDA #$40     Flag für 'BAM ungültig'
D884    8D F9 02    STA $02F9    setzen (da neu eingetragen wird)
D887    AD 4A 02    LDA $024A    aktuellen Dateityp holen
D88A    D0 1B       BNE $D8A7    liegt DEL-Datei vor ?
D88C    A9 02       LDA #$02     ja, Kennzeichen für PRG-Datei
D88E    8D 4A 02    STA $024A    als aktuellen Dateityp setzen
D891[1] AD 4A 02    LDA $024A    Dateityp holen
D894    D0 11       BNE $D8A7    ist als Typ 'DEL' angegeben ?
D896    A5 E7       LDA $E7      ja, Dateityp aus Kanaltabelle holen
D898    29 07       AND #$07     Flags ausblenden
D89A    8D 4A 02    STA $024A    und merken
D89D    AD 80 02    LDA $0280    Spurnummer des Sektors aus Puffertabelle
D8A0    D0 05       BNE $D8A7    ist Spur gesetzt ?
D8A2    A9 01       LDA #$01     nein, Kennzeichen für 'SEQ'
D8A4    8D 4A 02    STA $024A    in aktuellem Dateityp setzen
D8A7[3] AD 97 02    LDA $0297    Dateibetriebsart wieder holen
D8AA    C9 01       CMP #$01     und mit Wert für Schreiben vergleichen
```

```
D8AC     F0 18      BEQ $D8C6      soll Datei geschrieben werden ?
D8AE     4C 40 D9   JMP $D940      nein, Lesekanal öffnen
D8B1[1]  BC 7A 02   LDY $027A,X    Zeiger auf nächsten Parameter holen
D8B4     B9 00 02   LDA $0200,Y    und Zeichen des Parameters aus Eingabe-
D8B7     8D 58 02   STA $0258      puffer holen und speichern
D8BA     AD 80 02   LDA $0280      Spur des Datenblocks testen
D8BD     D0 B7      BNE $D876      ist Angabe gesetzt ?
D8BF     A9 01      LDA #$01       ja, Flag für Lesen/Schreiben
D8C1     8D 97 02   STA $0297      setzen
D8C4     D0 B0      BNE $D876      immer Sprung nach $D876
D8C6[1]  A5 E7      LDA $E7        Dateityp holen
D8C8     29 80      AND #$80       Flag für 'Joker vorhanden' holen
D8CA     AA         TAX            und merken
D8CB     D0 14      BNE $D8E1      ist im Dateinamen ein Joker ?
D8CD     A9 20      LDA #$20       nein, Flag für 'Datei nicht geschlossen'
D8CF     24 E7      BIT $E7        für ersten Namen testen
D8D1     F0 06      BEQ $D8D9      ist Datei geschlossen worden ?
D8D3     20 B6 C8   JSR $C8B6      nein, Dateieintrag in Directory löschen
D8D6     4C E3 D9   JMP $D9E3      neue Datei anlegen
D8D9[1]  AD 80 02   LDA $0280      Spurnummer des ersten Datenblocks
D8DC     D0 03      BNE $D8E1      ist Datei belegt ?
D8DE     4C E3 D9   JMP $D9E3      nein, neue Datei anlegen
D8E1[2]  AD 00 02   LDA $0200      erstes Zeichen aus Eingabepuffer holen
D8E4     C9 40      CMP #$40       und mit Replace-Befehl ('@') vergleichen
D8E6     F0 0D      BEQ $D8F5      bestehenden Eintrag überschreiben ?
D8E8     8A         TXA            Flag für Joker wieder holen
D8E9     D0 05      BNE $D8F0      ist Datei vorhanden ?
D8EB     A9 63      LDA #$63       ja, Fehlermeldung
D8ED     4C C8 C1   JMP $C1C8      '63 File Exists' ausgeben
D8F0[1]  A9 33      LDA #$33       Fehlermeldung
D8F2     4C C8 C1   JMP $C1C8      '33 Syntax Error' ausgeben
------------------------------------------------------------------------
[D8E6]
bestehenden Dateieintrag überschreiben
D8F5     A5 E7      LDA $E7        Dateityp des ersten Dateinamens holen
D8F7     29 07      AND #$07       und Flagbits ausblenden
D8F9     CD 4A 02   CMP $024A      mit aktuellem Dateityp vergleichen
D8FC     D0 67      BNE $D965      identisch ?
D8FE     C9 04      CMP #$04       ja, auf Wert für Relative Datei prüfen
D900     F0 63      BEQ $D965      liegt eine Relative Datei vor ?
D902     20 DA DC   JSR $DCDA      nein, Datei zum Schreiben öffnen
```

```
D905      A5 82     LDA $82        Nummer des geöffneten Kanals holen
D907      8D 70 02  STA $0270      und als aktuellen Schreibkanal merken
D90A      A9 11     LDA #$11       Kanalnummer für internen Lesekanal (17)
D90C      85 83     STA $83        als Sekundäradresse setzen
D90E      20 EB D0  JSR $D0EB      Kanal zum Lesen öffnen
D911      AD 94 02  LDA $0294      Position im akuellen Puffer holen
D914      20 C8 D4  JSR $D4C8      und Pufferadresse setzen
D917      A0 00     LDY #$00       Pufferzeiger initialisieren
D919      B1 94     LDA ($94),Y    Dateityp aus Directorypuffer holen
D91B      09 20     ORA #$20       Flagbit für Datei geöffnet setzen
D91D      91 94     STA ($94),Y    und wieder in Dateieintrag schreiben
D91F      A0 1A     LDY #$1A       Pufferzeiger auf Position der neuen
D921      A5 80     LDA $80        Spurnummer setzen; Spurnummer holen und
D923      91 94     STA ($94),Y    in Dateieintrag schreiben
D925      C8        INY            Pufferzeiger auf nächste Position
D926      A5 81     LDA $81        Nummer des aktuellen Sektors holen und
D928      91 94     STA ($94),Y    in Eintrag als ersetzende Werte merken
D92A      AE 70 02  LDX $0270      Nummer des aktuellen Schreibkanals holen
D92D      A5 D8     LDA $D8        Sektor des Dateieintrags holen
D92F      9D 60 02  STA $0260,X    und Nummer des Dateieintrags zuordnen
D932      A5 DD     LDA $DD        Zeiger auf Sektornummer im Eintrag holen
D934      9D 66 02  STA $0266,X    und Dateieintrag zuordnen
D937      20 3B DE  JSR $DE3B      aktuelle Spur- und Sektornummer an Job
D93A      20 64 D4  JSR $D464      Datensektor schreiben
D93D      4C EF D9  JMP $D9EF      Daten in Datei schreiben
D940 [1]  AD 80 02  LDA $0280      Spurnummer des ersten Eintrags holen
D943      D0 05     BNE $D94A      richtigen Eintrag gefunden ?
D945      A9 62     LDA #$62       nein, Fehlermeldung
D947      4C C8 C1  JMP $C1C8      '62 File Not Found ausgeben
D94A [1]  AD 97 02  LDA $0297      Dateibetriebsart feststellen
D94D      C9 03     CMP #$03       und mit Wert für 'M' vergleichen
D94F      F0 0B     BEQ $D95C      nicht geschlossene Datei lesen ?
D951      A9 20     LDA #$20       nein, Flag für 'Datei ist noch nicht
D953      24 E7     BIT $E7        ordnungsgemäß geschloßen' setzen
D955      F0 05     BEQ $D95C      ist Flag gesetzt ?
D957      A9 60     LDA #$60       ja, Fehlermeldung
D959      4C C8 C1  JMP $C1C8      '60 Write File Open' ausgeben
D95C [2]  A5 E7     LDA $E7        Dateityp holen und
D95E      29 07     AND #$07       reines Dateikennzeichen isolieren
D960      CD 4A 02  CMP $024A      mit aktuellem Dateityp vergleichen
D963      F0 05     BEQ $D96A      identisch ?
```

```
D965  A9 64     LDA #$64      nein, Fehlermeldung
D967  4C C8 C1  JMP $C1C8     '64 File Type Mismatch' ausgeben
D96A  A0 00     LDY #$00      Pufferzeiger
D96C  8C 79 02  STY $0279     zurücksetzen
D96F  AE 97 02  LDX $0297     Dateibetriebsart feststellen
D972  E0 02     CPX #$02      und mit Kennzeichen für 'A' vergleichen
D974  D0 1A     BNE $D990     sollen Daten angefügt werden ?
D976  C9 04     CMP #$04      ja, Dateityp auf Relative Datei prüfen
D978  F0 EB     BEQ $D965     liegt eine Relative Datei vor ?
D97A  B1 94     LDA ($94),Y   nein, Dateityp aus Directorypuffer
D97C  29 4F     AND #$4F      holen und Datei als geöffnet markieren
D97E  91 94     STA ($94),Y   Dateityp wieder in Eintrag zurück
D980  A5 83     LDA $83       aktuelle Sekundäradresse holen
D982  48        PHA           und retten
D983  A9 11     LDA #$11      Nummer des internen Lesekanals (17)
D985  85 83     STA $83       als aktuelle Sekundäradresse setzen
D987  20 3B DE  JSR $DE3B     aktuelle Spur und Sektor feststellen
D98A  20 64 D4  JSR $D464     Sektor auf Diskette schreiben
D98D  68        PLA           Sekundäradresse wieder holen
D98E  85 83     STA $83       und zurücksetzen
D990  20 A0 D9  JSR $D9A0     Datei zum Lesen öffnen
D993  AD 97 02  LDA $0297     Dateibetriebsart feststellen und mit
D996  C9 02     CMP #$02      Kennzeichen für 'A' (Append) vergleichen
D998  D0 55     BNE $D9EF     Daten an bestehende Datei anhängen ?
D99A  20 2A DA  JSR $DA2A     ja, Datei zum Anfügen vorbereiten
D99D  4C 94 C1  JMP $C194     und 'Ok' Meldung bereitstellen
-----------------------------------------------------------------------------

[CA26/D990]
Datei zum Lesen öffnen
D9A0  A0 13     LDY #$13      Zeiger auf Side-Sektor-Eintrag richten
D9A2  B1 94     LDA ($94),Y   Spur des ersten Side-Sektors holen
D9A4  8D 59 02  STA $0259     und merken
D9A7  C8        INY           Pufferzeiger auf nächstes Byte
D9A8  B1 94     LDA ($94),Y   Sektornummer des ersten Side-Sektors
D9AA  8D 5A 02  STA $025A     holen und merken
D9AD  C8        INY           Pufferzeiger auf nächste Position
D9AE  B1 94     LDA ($94),Y   Länge eines Records feststellen
D9B0  AE 58 02  LDX $0258     letzte Recordlänge holen
D9B3  8D 58 02  STA $0258     neue Recordlänge setzen
D9B6  8A        TXA           letzte Recordlänge
D9B7  F0 0A     BEQ $D9C3     nicht gesetzt ?
```

```
D9B9   CD 58 02   CMP $0258     nein, mit aktueller Länge vergleichen
D9BC   F0 05      BEQ $D9C3     letzter Record erreicht ?
D9BE   A9 50      LDA #$50      ja, Fehlermeldung
D9C0   20 C8 C1   JSR $C1C8     '50 Record Not Present' ausgeben
D9C3[2] AE 79 02  LDX $0279     Nummer des Dateinennung (0)
D9C6   BD 80 02   LDA $0280,X   aktuelle Spurnummer holen und
D9C9   85 80      STA $80       setzen
D9CB   BD 85 02   LDA $0285,X   aktuelle Sektornummer holen
D9CE   85 81      STA $81       und setzen
D9D0   20 46 DC   JSR $DC46     Lesekanal öffnen
D9D3   A4 82      LDY $82       Nummer des Kanals holen
D905   AE 79 02   LDX $0279     Nummer der verarbeiteten Datei
D9D8   B5 D8      LDA $D8,X     Sektornummer holen und
D9DA   99 60 02   STA $0260,Y   übertragen
D9DD   B5 DD      LDA $DD,X     Position in Dateieintrag holen
D9DF   99 66 02   STA $0266,Y   und übertragen
D9E2   60         RTS           zurück zur aufrufenden Routine
-----------------------------------------------------------------
[D8D6/D8DE]
Datei zum Schreiben öffnen
D9E3   A5 E2      LDA $E2       Nummer des angesprochenen Laufwerks
D9E5   29 01      AND #$01      herstellen
D9E7   85 7F      STA $7F       und als aktuelles Laufwerk merken
D9E9   20 DA DC   JSR $DCDA     Kanal zum Schreiben öffnen
D9EC   20 E4 D6   JSR $D6E4     Datei ins Directory eintragen
D9EF[2] A5 83     LDA $83       aktuelle Sekundäradresse holen
D9F1   C9 02      CMP #$02      und auf Beginn der Dateikanäle prüfen
D9F3   B0 11      BCS $DA06     soll Load oder Save ausgeführt werden ?
D9F5   20 3E DE   JSR $DE3E     ja, Spur und Sektor von Job holen
D9F8   A5 80      LDA $80       Spurnummer als letzte Spur des letzten
D9FA   85 7E      STA $7E       Zugriffs merken
D9FC   A5 7F      LDA $7F       Nummer des aktuellen Laufwerks holen
D9FE   8D 6E 02   STA $026E     und als letztes aktives Laufwerk merken
DA01   A5 81      LDA $81       Nummer des aktuellen Sektors als letzten
DA03   8D 6F 02   STA $026F     zugegriffenen Sektor merken
DA06[1] 4C 99 C1  JMP $C199     'Ok' Meldung bereitstellen
-----------------------------------------------------------------
[D866/D873]
Dateityp und Dateibetriebsart aus Befehlsstring feststellen
DA09   BC 7A 02   LDY $027A,X   Position des ersten Parameters holen
DA0C   B9 00 02   LDA $0200,Y   Zeichen aus Eingabepuffer holen
```

```
DA0F  A0 04     LDY #$04     Zähler auf Zahl der Betriebsarten setzen
DA11  88        DEY          Zähler auf nächstes Kennzeichen richten
DA12  30 08     BMI $DA1C    schon alle Betriebsarten geprüft ?
DA14  D9 B2 FE  CMP $FEB2,Y  nein, mit Dateibetriebsart vergleichen
DA17  D0 F8     BNE $DA11    identisch ?
DA19  8C 97 02  STY $0297    ja, Position in Eingabestring merken
DA1C  A0 05     LDY #$05     und Zähler für Dateitypen setzen
DA1E  88        DEY          Zähler auf nächsten Dateityp richten
DA1F  30 08     BMI $DA29    schon alle Dateitypen geprüft ?
DA21  D9 B6 FE  CMP $FEB6,Y  nein, mit Dateityp vergleichen
DA24  D0 F8     BNE $DA1E    identisch ?
DA26  8C 4A 02  STY $024A    ja, Position merken
DA29  60        RTS          zurück zur aufrufenden Routine
----------------------------------------------------------------------
[C996/D99A/DA32]
Datei für Append vorbereiten
DA2A  20 39 CA  JSR $CA39    Datenbyte lesen
DA2D  A9 80     LDA #$80     Flag für 'EOI aufgetreten'
DA2F  20 A6 DD  JSR $DDA6    testen
DA32  F0 F6     BEQ $DA2A    wurde letztes Byte gelesen ?
DA34  20 95 DE  JSR $DE95    ja, aktuelle Spur und Sektor feststellen
DA37  A6 81     LDX $81      Zeiger auf gültigen Datenbereich holen
DA39  E8        INX          und um eins erhöhen
DA3A  8A        TXA          (bei $FF wird er dann 0)
DA3B  D0 05     BNE $DA42    ist der Sektor ganz gefüllt ?
DA3D  20 A3 D1  JSR $D1A3    ja, Sektor auf Diskette schreiben
DA40  A9 02     LDA #$02     Pufferzeiger auf Beginn
DA42  20 C8 D4  JSR $D4C8    des Datenbereichs setzen
DA45  A6 82     LDX $82      Nummer des aktuellen Kanals holen
DA47  A9 01     LDA #$01     Flag für Schreiben
DA49  95 F2     STA $F2,X    in Kanalstatustabelle setzen
DA4B  A9 80     LDA #$80     Flag für Schreiben
DA4D  05 82     ORA $82      mit Kanalnummer verbinden
DA4F  A6 83     LDX $83      aktuelle Sekundäradresse holen und
DA51  9D 2B 02  STA $022B,X  Statusbyte der Sekundäradresse zuordnen
DA54  60        RTS          zurück zur aufrufenden Routine
----------------------------------------------------------------------
[D7FC]
Directory an Rechner übermitteln
DA55  A9 0C     LDA #$0C     Befehlsnummer 12
DA57  8D 2A 02  STA $022A    setzen
```

```
DA5A     A9 00      LDA #$00       Laufwerksnummer
DA5C     AE 74 02   LDX $0274      Länge des Befehlsstrings holen
DA5F     CA         DEX            und mit 1 vergleichen
DA60     F0 0B      BEQ $DA6D      hat der Befehl nur ein Zeichen ?
DA62     CA         DEX            nein, mit 2 vergleichen
DA63     D0 21      BNE $DA86      hat der Befehl nur zwei Zeichen ?
DA65     AD 01 02   LDA $0201      ja, zweites Zeichen holen und für
DA68     20 BD C3   JSR $C3BD      Laufwerksnummer überprüfen
DA6B     30 19      BMI $DA86      ist Laufwerksnennung fehlerfrei ?
DA6D[1]  85 E2      STA $E2        ja, Laufwerk merken
DA6F     EE 77 02   INC $0277      Zeiger auf ersten
DA72     EE 78 02   INC $0278      und auf den zweiten Parameter im
DA75     EE 7A 02   INC $027A      Befehlsstring auf nächste Position
DA78     A9 80      LDA #$80       Flag für 'Datei richtig geschlossen'
DA7A     85 E7      STA $E7        setzen
DA7C     A9 2A      LDA #$2A       '*' Joker als Dateiname in
DA7E     8D 00 02   STA $0200      Befehlsstring
DA81     8D 01 02   STA $0201      schreiben
DA84     D0 18      BNE $DA9E      immer Sprung nach $DA9E
DA86[2]  20 E5 C1   JSR $C1E5      ':' in Befehlstring suchen
DA89     D0 05      BNE $DA90      Doppelpunkt gefunden ?
DA8B     20 DC C2   JSR $C2DC      ja, Zeiger für Befehlsstring löschen
DA8E     A0 03      LDY #$03       Position des ersten Dateinamens (1)
DA90[1]  88         DEY            Zeiger auf Position des Namens
DA91     88         DEY            setzen
DA92     8C 7A 02   STY $027A      Zeiger auf ersten Dateinahmen setzen
DA95     20 00 C2   JSR $C200      Zeiger für Parameteranalyse setzen
DA98     20 98 C3   JSR $C398      Zeiger auf Dateinamen und Dateityp
DA9B     20 20 C3   JSR $C320      Laufwerksnummer aus Befehlsstring holen
DA9E[1]  20 CA C3   JSR $C3CA      Laufwerke für Zugriff vorbereiten
DAA1     20 B7 C7   JSR $C7B7      Directorytitel erzeugen
DAA4     20 9D C4   JSR $C49D      Dateinamen aus Directory holen
DAA7     20 9E EC   JSR $EC9E      Directoryzeile herstellen
DAAA     20 37 D1   JSR $D137      erstes Byte des Directory aus Puffer
DAAD     A6 82      LDX $82        Nummer des aktuellen Kanals
DAAF     9D 3E 02   STA $023E,X    Byte zur Ausgabe bereitstellen
DAB2     A5 7F      LDA $7F        aktuelles Laufwerk als Laufwerk, auf
DAB4     8D 8E 02   STA $028E      das zuletzt zugegriffen wurde merken
DAB7     09 04      ORA #$04       Flag für PRG-Datei setzen
DAB9     95 EC      STA $EC,X      und in Tabelle dem Kanal zuordnen
DABB     A9 00      LDA #$00       Zeiger in Eingabepuffer wieder
```

```
DABD   85 A3      STA $A3        bereit machen
DABF   60         RTS            zurück zur aufrufenden Routine
-----------------------------------------------------------------
[8151/9193/91A3/91B9/A9F1/BF60/E8CE]
Datei schließen
DAC0   A9 00      LDA #$00       Flag für 'BAM ungültig'
DAC2   8D F9 02   STA $02F9      setzen
DAC5   A5 83      LDA $83        aktuelle Sekundäradresse holen
DAC7   D0 0B      BNE $DAD4      Load-Befehl ?
DAC9   A9 00      LDA #$00       ja, Flag für 'Directory wird ausgegeben'
DACB   8D 54 02   STA $0254      löschen
DACE   20 27 D2   JSR $D227      Kanal schließen
DAD1[1] 4C DA D4  JMP $D4DA      interne Lese-/Schreibkanäle schließen
DAD4[1] C9 0F     CMP #$0F       Sekundäradresse mit 15 vergleichen
DAD6   F0 14      BEQ $DAEC      ist der Kommandokanal angesprochen ?
DAD8   20 02 DB   JSR $DB02      nein, Datei abschließen
DADB   A5 83      LDA $83        aktuelle Sekundäradresse holen
DADD   C9 02      CMP #$02       mit Beginn der Dateikanäle vergleichen
DADF   90 F0      BCC $DAD1      liegt Load/Save - Kanal vor (0/1) ?
DAE1   AD 6C 02   LDA $026C      nein, Fehlerflag holen und prüfen
DAE4   D0 03      BNE $DAE9      ist Fehler aufgetreten ?
DAE6   4C 94 C1   JMP $C194      nein, 'Ok' Meldung ausgeben
DAE9[1] 4C AD C1  JMP $C1AD      Fehlermeldung ausgeben
-----------------------------------------------------------------
[DAD6]
Alle Dateien abschließen
DAEC   A9 0E      LDA #$0E       höchste Sekundäradrese für Dateien
DAEE   85 83      STA $83        als aktuelle Sekundäradresse setzen
DAF0[1] 20 02 DB  JSR $DB02      Datei schließen
DAF3   C6 83      DEC $83        nächste Sekundäradresse anwählen
DAF5   10 F9      BPL $DAF0      schon alle Kanäle geschlossen ?
DAF7   AD 6C 02   LDA $026C      ja, Fehlerflag holen und prüfen
DAFA   D0 03      BNE $DAFF      ist Schließen fehlerfrei verlaufen ?
DAFC   4C 94 C1   JMP $C194      ja, 'Ok' Meldung ausgeben
DAFF[1] 4C AD C1  JMP $C1AD      Fehlermeldung ausgeben
-----------------------------------------------------------------
[C9F7/DADB/DAF0]
Datei, die durch Sekundäradresse bestimmt wird, abschließen
DB02   A6 83      LDX $83        aktuelle Sekundäradresse holen
DB04   BD 2B 02   LDA $022B,X    und entsprechenden Status feststellen
DB07   C9 FF      CMP #$FF       mit Wert für 'Kanal frei' vergleichen
```

```
DB09    D0 01      BNE $DB0C     ist Kanal belegt ?
DB0B    60         RTS           nein, zurück zur aufrufenden Routine
DB0C[1] 29 0F      AND #$0F      reine Kanalnummer herstellen
DB0E    85 82      STA $82       und merken
DB10    20 25 D1   JSR $D125     Dateityp holen und mit Kennzeichen
DB13    C9 07      CMP #$07      für Direktzugriffsdatei vergleichen
DB15    F0 0F      BEQ $DB26     identisch ?
DB17    C9 04      CMP #$04      nein, auf Wert für Relative Datei prüfen
DB19    F0 11      BEQ $DB2C     identisch ?
DB1B    20 07 D1   JSR $D107     nein, Kanal auf Schreibkanal prüfen
DB1E    B0 09      BCS $DB29     ist der Kanal zum Schreiben geöffnet ?
DB20    20 62 DB   JSR $DB62     ja, Schreiben zu Ende führen
DB23    20 A5 DB   JSR $DBA5     Eintrag im Directory schließen
DB26[1] 20 F4 EE   JSR $EEF4     BAM wieder auf Diskette zurückschreiben
DB29[1] 4C 27 D2   JMP $D227     Kanal schließen
DB2C[1] 20 F1 DD   JSR $DDF1     aktuellen Puffer auf Diskette schreiben
DB2F    20 1E CF   JSR $CF1E     neuen Puffer anlegen
DB32    20 CB E1   JSR $E1CB     Position des letzten Records holen
DB35    A6 D5      LDX $D5       Nummer des letzten Side-Sektors holen
DB37    86 73      STX $73       und merken
DB39    E6 73      INC $73       nächsten Side Sektor anwählen
DB3B    A9 00      LDA #$00      Zeropageadressen als
DB3D    85 70      STA $70       Zwischenspeicher
DB3F    85 71      STA $71       einrichten
DB41    A5 D6      LDA $D6       Position in Side-Sektor holen
DB43    38         SEC           und Zahl der Bytes für
DB44    E9 0E      SBC #$0E      Verkettung zu anderen Side-Sektoren
DB46    85 72      STA $72       berücksichtigen
DB48    20 51 DF   JSR $DF51     Zahl der Blöcke der Datei errechnen
DB4B    A6 82      LDX $82       Nummer des aktuellen Kanals
DB4D    A5 70      LDA $70       Zahl der Blocks der relativen Datei
DB4F    95 B5      STA $B5,X     (Low-Byte) in Tabelle übernehmen
DB51    A5 71      LDA $71       High-Byte
DB53    95 BB      STA $BB,X     kopieren
DB55    A9 40      LDA #$40      Flag für 'Eintrag korregieren' in
DB57    20 A6 DD   JSR $DDA6     Dateityp des Kanals prüfen
DB5A    F0 03      BEQ $DB5F     ist Flag in Dateityp gesetzt ?
DB5C    20 A5 DB   JSR $DBA5     ja, Directoryeintrag aktualisieren
DB5F[1] 4C 27 D2   JMP $D227     Kanal schließen
----------------------------------------------------------------------
```

```
[DB20]
Letzten Datensektor einer Datei auf Diskette schreiben
DB62     A6 82      LDX $82         Nummer des aktuellen Kanals holen
DB64     B5 B5      LDA $B5,X       dem Kanal zugeordnete Recordnummer
DB66     15 BB      ORA $BB,X       holen und prüfen
DB68     D0 0C      BNE $DB76       ist Recordnummer gesetzt ?
DB6A     20 E8 D4   JSR $D4E8       nein, aktuellen Pufferzeiger holen und
DB6D     C9 02      CMP #$02        mit Start des Datenbereichs vergleichen
DB6F     D0 05      BNE $DB76       ist der Sektor noch leer ?
DB71     A9 0D      LDA #$0D        ja, leeren Record ('Return') in
DB73     20 F1 CF   JSR $CFF1       Puffer schreiben
DB76[2]  20 E8 D4   JSR $D4E8       Pufferzeiger holen und mit Wert
DB79     C9 02      CMP #$02        für Start des Datenbereichs vergleichen
DB7B     D0 0F      BNE $DB8C       ist der Sektor noch leer ?
DB7D     20 1E CF   JSR $CF1E       ja, neuen Puffer anlegen
DB80     A6 82      LDX $82         Nummer des aktuellen Kanals feststellen
DB82     B5 B5      LDA $B5,X       zugeordnete Recordnummer (Low-Byte)
DB84     D0 02      BNE $DB88       ist Low-Byte auf Null ?
DB86     D6 BB      DEC $BB,X       ja, dann High-Byte der Recordnummer -1
DB88[1]  D6 B5      DEC $B5,X       Low-Byte um eins vermindern
DB8A     A9 00      LDA #$00        Rechenwert bei 'Puffer voll'
DB8C[1]  38         SEC             Anzahl der gültigen
DB8D     E9 01      SBC #$01        Datenbytes des Sektors berechnen
DB8F     48         PHA             und merken
DB90     A9 00      LDA #$00        Pufferzeiger auf Verkettungsbytes
DB92     20 C8 D4   JSR $D4C8       setzen
DB95     20 F1 CF   JSR $CFF1       Kennzeichen für letzten Sektor schreiben
DB98     68         PLA             Zahl der gültigen Datenbytes holen
DB99     20 F1 CF   JSR $CFF1       und in Sektor schreiben
DB9C     20 C7 D0   JSR $D0C7       Sektor auf Diskette zurückschreiben
DB9F     20 99 D5   JSR $D599       und warten bis Job ausgeführt ist
DBA2     4C 1E CF   JMP $CF1E       neuen Puffer anlegen
------------------------------------------------------------------------
[DB23/DB5C]
Directoryeintrag nach dem Schreiben abschließen
DBA5     A6 82      LDX $82         Nummer des aktuellen Kanals
DBA7     8E 70 02   STX $0270       holen und retten
DBAA     A5 83      LDA $83         Nummer der aktuellen Sekundäradresse
DBAC     48         PHA             holen und retten
DBAD     BD 60 02   LDA $0260,X     Nummer des Directorysektors für Eintrag
DBB0     85 81      STA $81         holen und als aktuellen Sektor setzen
```

```
DBB2  BD 66 02  LDA $0266,X   Position des Eintrags im Directory holen
DBB5  8D 94 02  STA $0294     und als aktuellen Pufferzeiger setzen
DBB8  B5 EC     LDA $EC,X     Dateityp des Kanals holen
DBBA  29 01     AND #$01      Nummer des Laufwerks feststellen
DBBC  85 7F     STA $7F       und als aktuelles Laufwerk übernehmen
DBBE  AD 85 FE  LDA $FE85     Nummer der Directoryspur holen
DBC1  85 80     STA $80       und als aktuelle Spur festlegen
DBC3  20 93 DF  JSR $DF93     Nummer des Puffers holen
DBC6  48        PHA           und merken
DBC7  85 F9     STA $F9       aktuelle Puffernummer setzen
DBC9  20 60 D4  JSR $D460     Directorysektor in Puffer einlesen
DBCC  A0 00     LDY #$00      Positionszeiger zurücksetzen
DBCE  BD E0 FE  LDA $FEE0,X   Pufferadresse (High-Byte) holen und als
DBD1  85 87     STA $87       High-Byte des Pufferzeigers übernehmen
DBD3  AD 94 02  LDA $0294     aktuelle Position im Puffer holen und
DBD6  85 86     STA $86       als Low-Byte übernehmen
DBD8  B1 86     LDA ($86),Y   Dateityp aus Directoryeintrag holen
DBDA  29 20     AND #$20      und Flag für 'Datei offen' prüfen
DBDC  F0 43     BEQ $DC21     ist die Datei schon geschlossen ?
DBDE  20 25 D1  JSR $D125     nein, Dateityp weiter prüfen
DBE1  C9 04     CMP #$04      und auf Wert für Relative Datei prüfen
DBE3  F0 44     BEQ $DC29     identisch ?
DBE5  B1 86     LDA ($86),Y   nein, gesamten Dateitypzeiger holen
DBE7  29 8F     AND #$8F      Flags löschen
DBE9  91 86     STA ($86),Y   und Dateityp wieder zurück in Eintrag
DBEB  C8        INY           Pufferzeiger auf nächste Position
DBEC  B1 86     LDA ($86),Y   Spurnummer des ersten Sektors der Datei
DBEE  85 80     STA $80       holen und als aktuelle Spur merken
DBF0  84 71     STY $71       aktuellen Pufferzeiger merken
DBF2  A0 1B     LDY #$1B      Pufferzeiger auf Sektor vom
DBF4  B1 86     LDA ($86),Y   Überschreiben setzen und Nummer holen
DBF6  48        PHA           Sektornummer merken
DBF7  88        DEY           Pufferzeiger auf dazugehörige Spur
DBF8  B1 86     LDA ($86),Y   setzen und Spurnummer holen
DBFA  D0 0A     BNE $DC06     ist kein Überschreibeintrag gesetzt ?
DBFC  85 80     STA $80       ja, Spurnummer und
DBFE  68        PLA           Sektornummer wieder holen und in
DBFF  85 81     STA $81       aktuelle Zeiger setzen
DC01  A9 67     LDA #$67      Fehlermeldung
DC03  20 45 E6  JSR $E645     '67 Illegal Track Or Sector' ausgeben
DC06  48        PHA           Spurnummer merken
```

```
DC07  A9 00     LDA #$00      Eintrag der Spur-
DC09  91 86     STA ($86),Y   und Sektornummer
DC0B  C8        INY           der überschreibenden Datei
DC0C  91 86     STA ($86),Y   löschen
DC0E  68        PLA           Spurnummer wieder holen
DC0F  A4 71     LDY $71       Pufferzeiger wieder zurücksetzen
DC11  91 86     STA ($86),Y   Spur des ersten Sektors der Datei setzen
DC13  C8        INY           Pufferzeiger auf nächstes Byte
DC14  B1 86     LDA ($86),Y   Nummer des alten Sektors holen
DC16  85 81     STA $81       und merken
DC18  68        PLA           Nummer des ersten Sektors der Datei
DC19  91 86     STA ($86),Y   holen und in Eintrag speichern
DC1B  20 7D C8  JSR $C87D     alte Dateisektoren löschen
DC1E  4C 29 DC  JMP $DC29     Datei schließen
DC21[1] B1 86   LDA ($86),Y   Dateityp aus Eintrag holen
DC23  29 0F     AND #$0F      reines Dateikennzeichen isolieren
DC25  09 80     ORA #$80      Flag für 'Datei geschlossen' setzen
DC27  91 86     STA ($86),Y   und als neuen Dateityp setzen
DC29[2] AE 70 02 LDX $0270    Nummer des aktuellen Kanals wieder holen
DC2C  A0 1C     LDY #$1C      Pufferzeiger auf Blockangabe setzen (28)
DC2E  B5 B5     LDA $B5,X     Zahl der Blocks der Datei (Low-Byte)
DC30  91 86     STA ($86),Y   holen und in Eintrag schreiben
DC32  C8        INY           Pufferzeiger auf nächstes Byte setzen
DC33  B5 BB     LDA $BB,X     High-Byte der Blockzahl holen und
DC35  91 86     STA ($86),Y   in Eintrag schreiben
DC37  68        PLA           aktuelle Puffernummer zurückholen
DC38  AA        TAX           und merken
DC39  A9 90     LDA #$90      Jobcode für 'Sektor schreiben'
DC3B  05 7F     ORA $7F       aktuelles Laufwerk in Jobcode eintragen
DC3D  20 90 D5  JSR $D590     Job ausführen
DC40  68        PLA           aktuelle Sekundäradresse wieder holen
DC41  85 83     STA $83       und zurücksetzen
DC43  4C 07 D1  JMP $D107     Kanalnummer holen
-------------------------------------------------------------------
[D47E/D7E4/D80B/D9D0/DC98:DD8A]
Kanal zum Lesen einer Datei öffnen
DC46  A9 01     LDA #$01      Zahl der Puffer
DC48  20 E2 D1  JSR $D1E2     Kanal zum Lesen öffnen
DC4B  20 B6 DC  JSR $DCB6     Zeiger des Kanals setzen
DC4E  AD 4A 02  LDA $024A     aktuellen Dateityp holen
DC51  48        PHA           und merken
```

```
DC52  0A        ASL A        Dateitypeintrag für Kanaltabelle
DC53  05 7F     ORA $7F      herstellen; Laufwerk einblenden
DC55  95 EC     STA $EC,X    und in Tabelle dem Kanal zuordnen
DC57  20 9B D0  JSR $D09B    Sektor in Puffer einlesen
DC5A  A6 82     LDX $82      aktuelle Kanalnummer holen
DC5C  A5 80     LDA $80      aktuelle Spurnummer feststellen
DC5E  D0 05     BNE $DC65    ist weiterer Sektor vorhanden ?
DC60  A5 81     LDA $81      nein, Zahl der gültigen Datenbytes
DC62  9D 44 02  STA $0244,X  holen und merken
DC65  68        PLA          Dateityp wieder zurückholen und mit
DC66  C9 04     CMP #$04     Wert für Relative Datei vergleichen
DC68  D0 3F     BNE $DCA9    liegt eine Relative Datei vor ?
DC6A  A4 83     LDY $83      ja, derzeitige Sekundäradresse holen
DC6C  B9 2B 02  LDA $022B,Y  zugeordnete Kanalnummer holen
DC6F  09 40     ORA #$40     Flag für Lesen
DC71  99 2B 02  STA $022B,Y  Sekundäradresse zuordnen
DC74  AD 58 02  LDA $0258    Länge des Records holen
DC77  95 C7     STA $C7,X    und dem Kanal zuordnen
DC79  20 8E D2  JSR $D28E    neuen Puffer für Side-Sektor belegen
DC7C  10 03     BPL $DC81    noch ein Puffer frei ?
DC7E  4C 0F D2  JMP $D20F    nein, Fehler '70 No Channel' ausgeben
DC81  A6 82     LDX $82      Nummer des aktuellen Kanals
DC83  95 CD     STA $CD,X    Puffernummer merken
DC85  AC 59 02  LDY $0259    Spurnummer des ersten Side-Sektors holen
DC88  84 80     STY $80      und als aktuelle Spurnummer setzen
DC8A  AC 5A 02  LDY $025A    Nummer des Sektors holen
DC8D  84 81     STY $81      und als aktuellen Sektor speichern
DC8F  20 D3 D6  JSR $D6D3    Spur- und Sektor an Jobschleife
DC92  20 73 DE  JSR $DE73    Side-Sektor von Diskette in Puffer lesen
DC95  20 99 D5  JSR $D599    warten bis Job ausgeführt
DC98  A6 82     LDX $82      Nummer des aktuellen Kanals holen
DC9A  A9 02     LDA #$02     und Recordzeiger in Tabelle
DC9C  95 C1     STA $C1,X    initialisieren
DC9E  A9 00     LDA #$00     Zeiger auf Beginn des Sektors
DCA0  20 C8 D4  JSR $D4C8    Pufferzeiger setzen
DCA3  20 53 E1  JSR $E153    ersten Record lesen und bereitstellen
DCA6  4C 3E DE  JMP $DE3E    Byte für Ausgabe und Zeiger setzen
DCA9  20 56 D1  JSR $D156    Byte aus Puffer holen
DCAC  A6 82     LDX $82      aktuelle Kanalnummer holen
DCAE  9D 3E 02  STA $023E,X  Byte für Ausgabe übergeben
DCB1  A9 88     LDA #$88     Ausgabeflag in Tabelle
```

```
DCB3   95 F2      STA $F2,X        mit Kanalstatus setzen
DCB5   60         RTS              zurück zur aufrufenden Routine
----------------------------------------------------------------------
[DC4B/DCE5]
Zeiger zum Kanal öffnen initialisieren
DCB6   A6 82      LDX $82          Nummer des Kanals holen
DCB8   B5 A7      LDA $A7,X        und zugeordeten Puffer feststellen
DCBA   0A         ASL A            Flag für Pufferbelegung prüfen
DCBB   30 06      BMI $DCC3        Puffer belegt ?
DCBD   A8         TAY              nein,
DCBE   A9 02      LDA #$02         Pufferzeiger auf Start des Datenbereichs
DCC0   99 99 00   STA $0099,Y      (Byte $02) setzen
DCC3[1] B5 AE     LDA $AE,X        Pufferstatus holen
DCC5   09 80      ORA #$80         und Flag für Puffer inaktiv
DCC7   95 AE      STA $AE,X        setzen
DCC9   0A         ASL A            Bit6 prüfen
DCCA   30 06      BMI $DCD2        soll Puffer zurückgeschrieben werden ?
DCCC   A8         TAY              nein, Puffernummer merken
DCCD   A9 02      LDA #$02         Pufferzeiger (Low-Byte) auf
DCCF   99 99 00   STA $0099,Y      Start des Datenbereichs setzen
DCD2[1] A9 00     LDA #$00         Zahl der freien Blocks (Low-Byte)
DCD4   95 B5      STA $B5,X        löschen
DCD6   4C 7F A9   JMP $A97F        Zahl der freien Blocks löschen
DCD9   EA         NOP              unbenutztes Byte
----------------------------------------------------------------------
[D48A/D902/D9E9]
Kanal um Datei zu schreiben öffnen
DCDA   20 A9 F1   JSR $F1A9        freien Sektor in BAM suchen
DCDD   A9 01      LDA #$01         Zahl der benötigten Puffer
DCDF   20 DF D1   JSR $D1DF        Puffer belegen
DCE2   20 D0 D6   JSR $D6D0        Spur- und Sektor an Jobschleife
DCE5   20 B6 DC   JSR $DCB6        Kanalzeiger initialisieren
DCE8   A6 82      LDX $82          Nummer des derzeitigen Kanals
DCEA   AD 4A 02   LDA $024A        aktuellen Dateityp holen
DCED   48         PHA              und merken
DCEE   0A         ASL A            Laufwerksnummer in
DCEF   05 7F      ORA $7F          Dateityp übernehmen und
DCF1   95 EC      STA $EC,X        in Tabelle dem Kanal zuordnen
DCF3   68         PLA              ursprünglichen Dateityp holen
DCF4   C9 04      CMP #$04         und auf Wert für Relative Datei prüfen
DCF6   F0 05      BEQ $DCFD        liegt eine Relative Datei vor ?
```

```
DCF8   A9 01      LDA #$01      nein, Flag für Schreiben
DCFA   95 F2      STA $F2,X     setzen
DCFC   60         RTS           zurück zur aufrufenden Routine
DCFD   A4 83      LDY $83       aktuelle Sekundäradresse holen
DCFF   B9 2B 02   LDA $022B,Y   und Kanal zuordnen
DD02   29 3F      AND #$3F      Flagbits im Kanalstatus zurücksetzen
DD04   09 40      ORA #$40      Flag für Lesen
DD06   99 2B 02   STA $022B,Y   setzen
DD09   AD 58 02   LDA $0258     Länge eines Records holen
DD0C   95 C7      STA $C7,X     und merken
DD0E   20 8E D2   JSR $D28E     Puffer suchen
DD11   10 03      BPL $DD16     freien Puffer gefunden ?
DD13   4C 0F D2   JMP $D20F     nein, Fehler '70 No Channel' ausgeben
DD16   A6 82      LDX $82       Nummer des aktuellen Kanals holen
DD18   95 CD      STA $CD,X     und Puffer für Side-Sektor belegen
DD1A   20 C1 DE   JSR $DEC1     Pufferinhalt löschen
DD1D   20 1E F1   JSR $F11E     freien Sektor in BAM suchen
DD20   A5 80      LDA $80       Spurnummer des Sektors als Spur
DD22   8D 59 02   STA $0259     des ersten Side-Sektors speichern
DD25   A5 81      LDA $81       Nummer des Sektors als Sektornummer
DD27   8D 5A 02   STA $025A     für den ersten Side-Sektor merken
DD2A   A6 82      LDX $82       aktuelle Kanalnummer holen
DD2C   B5 CD      LDA $CD,X     Nummer des entsprechenden Puffers holen
DD2E   20 D3 D6   JSR $D6D3     Spur- und Sektornummer an Jobschleife
DD31   A9 00      LDA #$00      Pufferzeiger auf Anfang
DD33   20 E9 DE   JSR $DEE9     des Puffers setzen
DD36   A9 00      LDA #$00      Kennzeichen für letzten Sektor in
DD38   20 8D DD   JSR $DD8D     Puffer schreiben
DD3B   A9 11      LDA #$11      Zahl der gültigen Bytes des Side-Sektors
DD3D   20 8D DD   JSR $DD8D     in Puffer übernehmen (17)
DD40   A9 00      LDA #$00      Nummer des Side-Sektors
DD42   20 8D DD   JSR $DD8D     in Puffer übertragen
DD45   AD 58 02   LDA $0258     Recordlänge in Side-Sektor
DD48   20 8D DD   JSR $DD8D     eintragen
DD4B   A5 80      LDA $80       aktuelle Spurnummer in
DD4D   20 8D DD   JSR $DD8D     Side-Sektor speichern
DD50   A5 81      LDA $81       aktuelle Sektornummer in
DD52   20 8D DD   JSR $DD8D     Side-Sektor speichern
DD55   A9 10      LDA #$10      Pufferzeiger auf Recorddaten
DD57   20 E9 DE   JSR $DEE9     im Side-Sektor setzen
DD5A   20 3E DE   JSR $DE3E     Spur und Sekktor des letzten Jobs holen
```

```
DD5D  A5 80      LDA $80        Spurnummer des ersten Datensektors
DD5F  20 8D DD   JSR $DD8D      in Side-Sektor schreiben
DD62  A5 81      LDA $81        Nummer des ersten Datensektors
DD64  20 8D DD   JSR $DD8D      in Side-Sektor übernehmen
DD67  20 6C DE   JSR $DE6C      Side-Sektor auf Diskette schreiben
DD6A  20 99 D5   JSR $D599      warten bis Job ausgeführt ist
DD6D  A9 02      LDA #$02       aktuellen Pufferzeiger auf
DD6F  20 C8 D4   JSR $D4C8      Beginn des Datenbereichs setzen
DD72  A6 82      LDX $82        Nummer des aktuellen Kanals
DD74  38         SEC            holen
DD75  A9 00      LDA #$00       Akku initialisieren und
DD77  F5 C7      SBC $C7,X      aus Recordlänge die Position des
DD79  95 C1      STA $C1,X      nächsten Records errechnen und setzen
DD7B  20 E2 E2   JSR $E2E2      Records in Sektor anlegen
DD7E  20 19 DE   JSR $DE19      Verkettung setzen
DD81  20 5E DE   JSR $DE5E      Sektor auf Diskette schreiben
DD84  20 99 D5   JSR $D599      und warten bis Job ausgeführt ist
DD87  20 F4 EE   JSR $EEF4      neue BAM auf Diskette schreiben
DD8A  4C 98 DC   JMP $DC98      Rückmeldung ausgeben
-----------------------------------------------------------------
[DD38/DD3D/DD42/DD48/DD4D/DD52/DD5F/DD64/E3FA/E3FE]
Ein Byte in aktuellen Side-Sektor schreiben
DD8D  48         PHA            Byte merken
DD8E  A6 82      LDX $82        Nummer des aktuellen Kanals holen
DD90  B5 CD      LDA $CD,X      und entsprechenden Puffer feststellen
DD92  4C FD CF   JMP $CFFD      Byte in Puffer übertragen
-----------------------------------------------------------------
[Einsprungstelle wird im DOS nicht genutzt]
Kanalnummer im Dateityp-Flag setzen (Carry=1) oder löschen (Carry=0)
DD95  90 06      BCC $DD9D      Flags löschen ?
-----------------------------------------------------------------
[CA4F/DD97/E01A/E0A0/E107/E25F]
Wert in Dateitypflag einblenden (Bit =1: setzen)
DD97  A6 82      LDX $82        nein, Nummer des aktuellen Kanals holen
DD99  15 EC      ORA $EC,X      und in Dateitypflag einblenden
DD9B  D0 06      BNE $DDA3      immer Sprung nach $DDA3
-----------------------------------------------------------------
[CFAC/DD95/DFD2/E003/E0ED/E21B]
Wert in Dateitypflag ausblenden (Bit =1: ausblenden)
DD9D  A6 82      LDX $82        aktuelle Kanalnummer holen
DD9F  49 FF      EOR #$FF       und invertieren
```

```
DDA1  35 EC     AND $EC,X       Nummer aus Dateitypflag ausblenden
DDA3  95 EC     STA $EC,X       neues Dateitypflag setzen
DDA5  60        RTS             zurück zur aufrufenden Routine
----------------------------------------------------------------------
[C9DD/DA2F/DB57/DFD7/E0AD/E0BE/E0F5/E122/E26A]
Dateitypflag auf gesetzt prüfen (Flag-Wert in Akku)
DDA6  A6 82     LDX $82         Nummer des aktuellen Kanals holen
DDA8  35 EC     AND $EC,X       und entsprechende Flags prüfen
DDAA  60        RTS             zurück zur aufrufenden Routine
----------------------------------------------------------------------
[CF4C/E052/E060]
Prüfen ob Jobcode für Schreiben vorliegt
DDAB  20 93 DF  JSR $DF93       Nummer des derzeitigen Puffers holen
DDAE  AA        TAX             und merken
DDAF  BD 5B 02  LDA $025B,X     letzten Jobcode des Puffers holen
DDB2  29 F0     AND #$F0        und reine Befehlsbits herstellen
DDB4  C9 90     CMP #$90        mit Wert für Schreiben vergleichen
DDB6  60        RTS             zurück zur aufrufenden Routine
----------------------------------------------------------------------
[C835]
Dateizeiger prüfen
DDB7  A2 00     LDX #$00        Sekundäradresse setzen
DDB9[1] 86 71   STX $71         und merken
DDBB  BD 2B 02  LDA $022B,X     zugehorige Kanalnummer holen
DDBE  C9 FF     CMP #$FF        mit Wert für 'Kanal frei' vergleichen
DDC0  D0 08     BNE $DDCA       ist Kanal belegt ?
DDC2[3] A6 71   LDX $71         ja, Nummer der Sekundäradresse wieder
DDC4  E8        INX             holen und nächste Adresse anwählen
DDC5  E0 10     CPX #$10        mit maximaler Adresse +1 vergleichen
DDC7  90 F0     BCC $DDB9       ist Sekundäradresse erlaubt ?
DDC9  60        RTS             nein, zurück zur aufrufenden Routine
DDCA[1] 86 71   STX $71         freie Sekundäradresse merken
DDCC  29 3F     AND #$3F        Kanalnummer feststellen
DDCE  A8        TAY             und merken
DDCF  B9 EC 00  LDA $00EC,Y     Dateitypflag holen und Nummer
DDD2  29 01     AND #$01        des angewählten Laufwerks holen
DDD4  85 70     STA $70         Laufwerksnummer merken
DDD6  AE 53 02  LDX $0253       Nummer des Eintrags
DDD9  B5 E2     LDA $E2,X       Standardlaufwerk des Kanals
DDDB  29 01     AND #$01        holen
DDDD  C5 70     CMP $70         und mit angewähltem Laufwerk vergleichen
```

```
DDDF  D0 E1     BNE $DDC2       identisch ?
DDE1  B9 60 02  LDA $0260,Y     ja, Nummer des Directorysektors holen
DDE4  D5 D8     CMP $D8,X       und mit Sektor des Eintrags vergleichen
DDE6  D0 DA     BNE $DDC2       identisch ?
DDE8  B9 66 02  LDA $0266,Y     ja, Position des Eintrags holen
DDEB  D5 DD     CMP $DD,X       und auf Position in Directory testen
DDED  D0 D3     BNE $DDC2       identisch ?
DDEF  18        CLC             ja, Flag für alle Zeiger ok
DDF0  60        RTS             zurück zur aufrufenden Routine
-----------------------------------------------------------------
[DB2C/E2AA/E454]
Puffer auf Diskette schreiben
DDF1  20 9E DF  JSR $DF9E       Status des Puffers prüfen
DDF4  50 06     BVC $DDFC       sind Daten im Puffer geändert worden
DDF6  20 5E DE  JSR $DE5E       ja, Sektor auf Diskette schreiben
DDF9  20 99 D5  JSR $D599       warten bis Job ausgeführt ist
DDFC  60        RTS             zurück zur aufrufenden Routine
-----------------------------------------------------------------
[E3AC/E3BF]
Verkettungsbytes, die auf nächsten Sektor zeigen, setzen
DDFD  20 2B DE  JSR $DE2B       aktuellen Pufferzeiger setzen
DE00  A5 80     LDA $80         Spurnummer des nächsten Sektors in
DE02  91 94     STA ($94),Y     Puffer übertragen
DE04  C8        INY             Pufferzeiger auf nächste Position
DE05  A5 81     LDA $81         Nummer des nächsten Sektors
DE07  91 94     STA ($94),Y     in aktuellen Puffer schreiben
DE09  4C 05 E1  JMP $E105       Puffer als 'geändert' kennzeichnen
-----------------------------------------------------------------
[E2AD/E3D7]
Verkettungsbytes, die auf nächsten Sektor zeigen, holen
DE0C  20 2B DE  JSR $DE2B       aktuellen Pufferzeiger setzen
DE0F  B1 94     LDA ($94),Y     Spurnummer des nächsten Sektors aus
DE11  85 80     STA $80         Puffer holen und merken
DE13  C8        INY             Pufferzeiger auf nächstes Byte setzen
DE14  B1 94     LDA ($94),Y     Nummer des nächsten Sektors aus Puffer
DE16  85 81     STA $81         holen und als aktuellen Sektor merken
DE18  60        RTS             zurück zur aufrufenden Routine
-----------------------------------------------------------------
```

```
[DD7E/E3D1]
Kennzeichen für letzten Sektor in Verkettungsbytes setzen
DE19  20 2B DE   JSR $DE2B      aktuellen Pufferzeiger setzen
DE1C  A9 00      LDA #$00       Kennzeichen für letzten Sektor in
DE1E  91 94      STA ($94),Y    Puffer schreiben
DE20  C8         INY            Pufferzeiger auf nächstes Byte
DE21  A6 82      LDX $82        Nummer des aktuellen Kanals holen
DE23  B5 C1      LDA $C1,X      Zahl der gültigen Datenbytes aus Tabelle
DE25  AA         TAX            holen und
DE26  CA         DEX            korregieren
DE27  8A         TXA            (da 0 mitgezählt wird)
DE28  91 94      STA ($94),Y    Zahl in Puffer schreiben
DE2A  60         RTS            zurück zur aufrufenden Routine
------------------------------------------------------------------
[DDFD/DE0C/DE19/E1B2/E2E2]
Pufferzeiger des aktuellen Puffers auf Null setzen
DE2B  20 93 DF   JSR $DF93      Nummer des Puffers holen
DE2E  0A         ASL A          und verdoppeln  (da Zeigertabelle aus
DE2F  AA         TAX            2-Byte-Werten besteht)
DE30  B5 9A      LDA $9A,X      Highbyte der Pufferadresse holen
DE32  85 95      STA $95        und in Pufferzeiger übernehmen
DE34  A9 00      LDA #$00       Low-Byte auf Start des Puffers
DE36  85 94      STA $94        setzen
DE38  A0 00      LDY #$00       Indexzeiger auf Anfang zurücksetzen
DE3A  60         RTS            zurück zur aufrufenden Routine
------------------------------------------------------------------
[C5E8/C634/D48D/D6F4/D937/D987]
aktuelle Spur- und Sektornummer des aktuellen Jobs holen
DE3B  20 EB D0   JSR $D0EB      Kanalnummer der Sekundäradresse holen
------------------------------------------------------------------
[D9F5/DCA6/DD5A/E2D0/E3E0/E824/E840/F11E]
Spur- und Sektornummer des aktuellen Jobs holen
DE3E  20 93 DF   JSR $DF93      zugehörige Puffernumer feststellen
DE41  85 F9      STA $F9        und als aktuellen Puffer merken
DE43  0A         ASL A          Puffernummer verdoppeln (da Spur/Sektor-
DE44  A8         TAY            Tabelle aus 2-Byte-Werten besteht)
DE45  B9 06 00   LDA $0006,Y    Spurnummer des aktuellen Jobs holen
DE48  85 80      STA $80        und als aktuelle Spurnummer merken
DE4A  B9 07 00   LDA $0007,Y    Nummer des dazugehörigen Sektors holen
DE4D  85 81      STA $81        und als aktuellen Sektor übernehmen
DE4F  60         RTS            zurück zur aufrufenden Routine
```

```
-------------------------------------------------------------------
[E4A6/DE57:CAA9,CF5A,E057,E065,E075/DE5E:C0BB,CAC3,DD81,DDF6,E047,E3B3]
[E3D4,E4ED/DE6C:DD67,E42D/DE73:DC92]
Jobcodes an Jobschleife übergeben
DE50  A9 90      LDA #$90      Jobcode für 'Sektor schreiben'
DE52  8D 4D 02   STA $024D     setzen
DE55  D0 28      BNE $DE7F     immer Sprung nach $DE7F
DE57  A9 80      LDA #$80      Jobcode für 'Sektor lesen'
DE59  8D 4D 02   STA $024D     setzen
DE5C  D0 21      BNE $DE7F     immer Sprung nach $DE7F
DE5E  A9 90      LDA #$90      Jobcode für 'Sektor schreiben'
DE60  8D 4D 02   STA $024D     setzen
DE63  D0 26      BNE $DE8B     immer Sprung nach $DE7F
DE65  A9 80      LDA #$80      Jobcode für Sektor lesen
DE67  8D 4D 02   STA $024D     setzen
DE6A  D0 1F      BNE $DE8B     immer Sprung nach $DE7F
DE6C  A9 90      LDA #$90      Jobcode für 'Sektor schreiben'
DE6E  8D 4D 02   STA $024D     setzen
DE71  D0 02      BNE $DE75     immer Sprung nach $DE7F
DE73  A9 80      LDA #$80      Jobcode für 'Sektor lesen'
DE75  8D 4D 02   STA $024D     setzen
DE78  A6 82      LDX $82       Nummer des aktuellen Kanals holen
DE7A  B5 CD      LDA $CD,X     Nummer des dritten Puffers feststellen
DE7C  AA         TAX           und merken
DE7D  10 13      BPL $DE92     ist Puffer benutzt ?
DE7F  20 D0 D6   JSR $D6D0     nein, Spur und Sektor an Jobschleife
DE82  20 93 DF   JSR $DF93     Nummer des aktuellen Puffers holen
DE85  AA         TAX           und merken
DE86  A5 7F      LDA $7F       Nummer des aktuellen Laufwerks holen
DE88  9D 5B 02   STA $025B,X   und Puffer zuordnen
DE8B  20 15 E1   JSR $E115     Flag für ' Puffer geändert' löschen
DE8E  20 93 DF   JSR $DF93     Nummer des aktuellen Puffers holen
DE91  AA         TAX           und merken
DE92  4C 06 D5   JMP $D506     Parameter prüfen und Job ausführen
-------------------------------------------------------------------
[DA34/E03F/E06B]
Parameter des nächsten Sektors anhand der Verkettungsbytes setzen
DE95  A9 00      LDA #$00      aktuellen Pufferzeiger auf
DE97  20 C8 D4   JSR $D4C8     Start des Puffers zurücksetzen
DE9A  20 37 D1   JSR $D137     Byte aus Puffer holen und
DE9D  85 80      STA $80       als aktuelle Spurnummer übernehmen
```

```
DE9F  20 37 D1   JSR $D137        Byte aus Puffer holen und
DEA2  85 81      STA $81          als Nummer des aktuellen Sektors setzen
DEA4  60         RTS              zurück zur aufrufenden Routine
-----------------------------------------------------------------
```

[E467]

Daten aus Puffer in anderen Puffer kopieren
(Akku muß Zahl der Bytes enthalten; Y die Nummer des Quellpuffers;
X enthält die Nummer des Zielpuffers)

```
DEA5  48         PHA              Zahl der zu kopierenden Bytes merken
DEA6  A9 00      LDA #$00         Low-Bytes der beiden
DEA8  85 6F      STA $6F          Pufferzeiger
DEAA  85 71      STA $71          löschen
DEAC  B9 E0 FE   LDA $FEE0,Y      High-Byte der Pufferadresse des Quell-
DEAF  85 70      STA $70          puffers setzen
DEB1  BD E0 FE   LDA $FEE0,X      High-Byte der Pufferadresse des Ziel-
DEB4  85 72      STA $72          puffers setzen
DEB6  68         PLA              Zahl der zu übertragenden Bytes
DEB7  A8         TAY              wieder holen und merken
DEB8  88         DEY              Zähler initialisieren
DEB9  B1 6F      LDA ($6F),Y      Byte aus Quellpuffer lesen und
DEBB  91 71      STA ($71),Y      in Zielpuffer übertragen
DEBD  88         DEY              Pufferzeiger auf nächstes Byte setzen
DEBE  10 F9      BPL $DEB9        schon alle Daten übertragen ?
DEC0  60         RTS              ja, zurück zur aufrufenden Routine
-----------------------------------------------------------------
```

[DD1A/E45B]

Puffer mit $00 löschen (Nummer in A)

```
DEC1  A8         TAY              Puffernummer merken
DEC2  B9 E0 FE   LDA $FEE0,Y      High-Byte der Pufferadresse holen
DEC5  85 70      STA $70          und in Zeiger festlegen
DEC7  A9 00      LDA #$00         Low-Byte des Zeigers
DEC9  85 6F      STA $6F          auf den Pufferbeginn setzen
DECB  A8         TAY              Puffer mit Wert der Puffernummer
DECC  91 6F      STA ($6F),Y      löschen
DECE  C8         INY              Pufferzeiger auf nächste Position
DECF  D0 FB      BNE $DECC        schon ganzer Puffer gelöscht ?
DED1  60         RTS              ja, zurück zur aufrufenden Routine
-----------------------------------------------------------------
```

```
[DF66/E1CB]
Nummer des aktuellen Side-Sektors holen
DED2  A9 00     LDA #$00       Pufferadresse holen
DED4  20 DC DE  JSR $DEDC      und in Zeiger $94/$95 setzen
DED7  A0 02     LDY #$02       Position im Puffer anwählen
DED9  B1 94     LDA ($94),Y    Nummer des Side-Sektors aus Sektor holen
DEDB  60        RTS            zurück zur aufrufenden Routine
------------------------------------------------------------------------
[DED4/DEEA/E41E/E46C]
Pufferzeiger $94/$95 auf beliebige Position in Puffer setzen
DEDC  85 94     STA $94        gewünschte Position im Puffer merken
DEDE  A6 82     LDX $82        Nummer des aktuellen Kanals holen
DEE0  B5 CD     LDA $CD,X      und zugeordneten 3. Puffer feststellen
DEE2  AA        TAX            und Puffernummer merken
DEE3  BD E0 FE  LDA $FEE0,X    High-Byte der Pufferadresse holen
DEE6  85 95     STA $95        und in Zeiger setzen
DEE8  60        RTS            zurück zur aufrufenden Routine
------------------------------------------------------------------------
[DD33/DD57/DF14/E1FF/E35A/E3F4]
Pufferzeiger setzen
DEE9  48        PHA            gewünschte Position im Puffer merken
DEEA  20 DC DE  JSR $DEDC      Pufferzeiger setzen
DEED  48        PHA            High-Byte der Pufferadresse merken
DEEE  8A        TXA            Nummer des aktuellen Puffers holen
DEEF  0A        ASL A          und verdoppeln (da Tabelle 2-Byte-Werte
DEF0  AA        TAX            enthalten)
DEF1  68        PLA            High-Byte der Pufferadresse wieder holen
DEF2  95 9A     STA $9A,X      und in Tabelle der Pufferadressen setzen
DEF4  68        PLA            Position im Puffer holen
DEF5  95 99     STA $99,X      und in Tabelle eintragen
DEF7  60        RTS            zurück zur aufrufenden Routine
------------------------------------------------------------------------
[E258/E43C]
Side-Sector in Puffer lesen und Zeiger setzen
DEF8  20 66 DF  JSR $DF66      Status des  Side-Sektors prüfen
DEFB  30 0E     BMI $DF0B      existiert Side-Sektor ?
DEFD  50 13     BVC $DF12      ja, ist Side-Sektor im Puffer ?
DEFF  A6 82     LDX $82        nein, Nummer des aktuellen Kanals holen
DF01  B5 CD     LDA $CD,X      zugeordnete Puffernummer feststellen
DF03  20 1B DF  JSR $DF1B      Side-Sektor in Puffer einlesen
DF06  20 66 DF  JSR $DF66      Status nochmals prüfen
```

```
DF09    10 07     BPL $DF12      ist alles fehlerfrei verlaufen ?
DF0B[1] 20 CB E1  JSR $E1CB      nein, Ende der rel. Datei suchen
DF0E    2C CE FE  BIT $FECE      N und V Prozessorflag setzen
DF11    60        RTS            zurück zur aufrufenden Routine
DF12[2] A5 D6     LDA $D6        Position in Side-Sektor holen
DF14    20 E9 DE  JSR $DEE9      und Pufferzeiger setzen
DF17    2C CD FE  BIT $FECD      alle Flags löschen
DF1A    60        RTS            zurück zur aufrufenden Routine
```

[DF03/E1EC/E4D6]

Sektor lesen (Pufferzeiger des aktuellen Puffer muß auf Spur- und Sektorparameter der Verkettungsbytes gerichtet sein)

```
DF1B    85 F9     STA $F9        Nummer des Puffers merken
DF1D    A9 80     LDA #$80       Jobcode für 'Sektor lesen' setzen
DF1F    D0 04     BNE $DF25      immer Sprung nach $DF25
DF21    85 F9     STA $F9        Nummer des aktuellen Puffers merken
DF23    A9 90     LDA #$90       Jobcode für 'Sektor schreiben' setzen
DF25[1] 48        PHA            und Jobcode merken
DF26    B5 EC     LDA $EC,X      Dateityp des Kanals holen
DF28    29 01     AND #$01       und angewähltes Laufwerk feststellen
DF2A    85 7F     STA $7F        als aktuelle Laufwerksnumer übernehmen
DF2C    68        PLA            Jobcode wieder holen
DF2D    05 7F     ORA $7F        und Laufwerksnummer mit einblenden
DF2F    8D 4D 02  STA $024D      Jobcode merken
DF32    B1 94     LDA ($94),Y    Nummer des nächsten Sektors aus Puffer
DF34    85 80     STA $80        lesen und speichern
DF36    C8        INY            Pufferzeiger auf nächstes Byte setzen
DF37    B1 94     LDA ($94),Y    Nummer des Sektors aus Puffer holen
DF39    85 81     STA $81        und übernehmen
DF3B    A5 F9     LDA $F9        Nummer des aktuellen Puffers
DF3D    20 D3 D6  JSR $D6D3      Spur- und Sektorparameter an Jobschleife
DF40    A6 F9     LDX $F9        Nummer des aktuellen Puffers holen
DF42    4C 93 D5  JMP $D593      Job ausführen
```

[E3E9/E40F/E418]

Zeiger für Side-Sektor setzen

```
DF45    A6 82     LDX $82        Nummer des aktuellen Kanals
DF47    B5 CD     LDA $CD,X      Nummer des zugeordneten Puffers holen
DF49    4C EB D4  JMP $D4EB      Pufferzeiger setzen
```

```
[DF52/Einsprung über DF51/DF5C in DB48/DF4E/DF57/E381/DF51:DB48]
Zahl der Sektoren einer Relativen Datei berechnen
DF4C     A9 78      LDA #$78      Zahl der Sektorzeiger pro Side-Sektor
DF4E     20 5C DF   JSR $DF5C     in Zähler addieren
DF51[1]  CA         DEX           nächste Side-Sektornummer setzen
DF52     10 F8      BPL $DF4C     alle Side-Sektoren berücksichtigt ?
DF54     A5 72      LDA $72       ja, Zahl der Verkettungsbytes
DF56     4A         LSR A         durch 2 teilen
DF57     20 5C DF   JSR $DF5C     dazu addieren
DF5A     A5 73      LDA $73       Zahl der Side-Sektoren
DF5C[2]  18         CLC           Addition initialisieren
DF5D     65 70      ADC $70       Wert zu Zähler
DF5F     85 70      STA $70       addieren
DF61     90 02      BCC $DF65     ist ein Übertrag aufgetreten ?
DF63     E6 71      INC $71       ja, High-Byte korregieren
DF65[1]  60         RTS           zurück zur aufrufenden Routine
-----------------------------------------------------------------
[DEF8/DF06]
Status eines Side-Sektors prüfen
DF66     20 D2 DE   JSR $DED2     Nummer des Side-Sektors aus Puffer holen
DF69     C5 D5      CMP $D5       mit gesuchtem Sektor vergleichen
DF6B     D0 0E      BNE $DF7B     ist der richtige Side-Sektor im Puffer ?
DF6D     A4 D6      LDY $D6       ja, Zeiger in Puffer holen
DF6F     B1 94      LDA ($94),Y   Spurnummer des Records holen
DF71     F0 04      BEQ $DF77     ist Record angelegt ?
DF73     2C CD FE   BIT $FECD     ja, Fehlerflags löschen
DF76     60         RTS           zurück zur aufrufenden Routine
DF77[1]  2C CF FE   BIT $FECF     Flag für 'kein Record' setzen
DF7A     60         RTS           zurück zur aufrufenden Routine
DF7B[1]  A5 D5      LDA $D5       Nummer des gesuchten Side-Sektors holen
DF7D     C9 06      CMP #$06      mit größtem Side-Sektor vergleichen
DF7F     B0 0A      BCS $DF8B     ist Nummer im erlaubten Bereich ?
DF81     0A         ASL A         ja, Side-Sektornummer verdoppeln
DF82     A8         TAY           und merken
DF83     A9 04      LDA #$04      Pufferzeiger setzen
DF85     85 94      STA $94       und speichern
DF87     B1 94      LDA ($94),Y   Spurnummer des Side-Sektors holen
DF89     D0 04      BNE $DF8F     ist Spur gesetzt ?
DF8B[1]  2C D0 FE   BIT $FED0     nein, Fehlerflags setzen
DF8E     60         RTS           zurück zur aufrufenden Routine
DF8F[1]  2C CE FE   BIT $FECE     Flag für 'Sektor nicht im Puffer' setzen
```

```
DF92   60          RTS              zurück zur aufrufenden Routine
-----------------------------------------------------------------------
[CAB7/CDEC/CF6F/CFF2/D0CC/D12F/D1D3/D324/D44D/D4CA/D4E8/D4F8/D6D0/D75E]
[DBC3/DDAB/DE2B/DE3E/DE82/DE8E/E07F/E457/E4D1/ECB3/ECDC/ED0D/ED32/ED46]
[EEF4]
Nummer des aktuellen Puffers feststellen
DF93   A6 82       LDX $82          Nummer des aktuellen Kanals holen
DF95   B5 A7       LDA $A7,X        Pufferbelegung prüfen
DF97   10 02       BPL $DF9B        ist Puffer belegt ?
DF99   B5 AE       LDA $AE,X        Nummer des zweiten Puffers
DF9B   29 BF       AND #$BF         holen
DF9D   60          RTS              zurück zur aufrufenden Routine
-----------------------------------------------------------------------
[DDF1/E042/E10A/E115/E4B1]
Status des akuellen Puffers holen
DF9E   A6 82       LDX $82          Nummer des derzeitigen Kanals holen
DFA0   8E 57 02    STX $0257        und merken
DFA3   B5 A7       LDA $A7,X        Puffernummer holen
DFA5   10 09       BPL $DFB0        ist Puffer belegt ?
DFA7   8A          TXA              ja, Kanalnummer wieder holen
DFA8   18          CLC              und auf Nummer
DFA9   69 07       ADC #$07         für Zugriff auf zweiten
DFAB   8D 57 02    STA $0257        Puffer umrechnen und merken
DFAE   B5 AE       LDA $AE,X        Pufferstatus holen und prüfen
DFB0 1 85 70       STA $70          Status merken
DFB2   29 1F       AND #$1F         Flags ausblenden
DFB4   24 70       BIT $70          ist Puffer aktiv ?
DFB6   60          RTS              zurück zur aufrufenden Routine
-----------------------------------------------------------------------
[CF21/CF7E]
Prüfen ob Puffer frei ist
DFB7   A6 82       LDX $82          Nummer des aktuellen Kanals
DFB9   B5 A7       LDA $A7,X        dazugehörige Puffernummer holen
DFBB   30 02       BMI $DFBF        ist Puffer belegt ?
DFBD   B5 AE       LDA $AE,X        ja, Pufferstatus prüfen
DFBF 1 C9 FF       CMP #$FF         mit Wert für 'Puffer aktiv' vergleichen
DFC1   60          RTS              zurück zur aufrufenden Routine
-----------------------------------------------------------------------
```

```
[CF2E/CF88]
aktuellen Puffer aktivieren (bei 2-Puffer-Betrieb)
DFC2  A6 82     LDX $82        Nummer des aktuellen Kanals holen
DFC4  09 80     ORA #$80       Flag für 'Puffer inaktiv' setzen
DFC6  B4 A7     LDY $A7,X      Nummer des passenden Puffers holen
DFC8  10 03     BPL $DFCD      ist 1. Puffer belegt ?
DFCA  95 A7     STA $A7,X      nein, 1. Puffer aktivieren
DFCC  60        RTS            zurück zur aufrufenden Routine
DFCD  95 AE     STA $AE,X      Nummer für 2. Puffer dem Kanal zuordnen
DFCF  60        RTS            zurück zur aufrufenden Routine
-----------------------------------------------------------------------
[E153/E009:E291]
Record einer Relativen Datei schreiben
DFD0  A9 20     LDA #$20       Flag für 'Record voll'
DFD2  20 9D DD  JSR $DD9D      löschen
DFD5  A9 80     LDA #$80       Flag für letztes Byte (EOI)
DFD7  20 A6 DD  JSR $DDA6      Flag prüfen
DFDA  D0 41     BNE $701D      ist letztes Byte empfangen worden ?
DFDC  A6 82     LDX $82        nein, Nummer des aktuellen Kanals holen
DFDE  F6 B5     INC $B5,X      und Nummer des Records erhöhen
DFE0  D0 02     BNE $DFE4      ist ein Übertrag emtstanden ?
DFE2  F6 BB     INC $BB,X      ja, High-Byte auch korregieren
DFE4  A6 82     LDX $82        aktuelle Kanalnummer holen
DFE6  B5 C1     LDA $C1,X      Zeiger auf Position im Puffer holen
DFE8  F0 2E     BEQ $7018      Zeiger gesetzt ?
DFEA  20 E8 D4  JSR $D4E8      ja, Pufferzeiger wieder holen
DFED  A6 82     LDX $82        Nummer des aktuellen Kanals holen und
DFEF  D5 C1     CMP $C1,X      Puffer- mit Recordzeiger vergleichen
DFF1  90 03     BCC $DFF6      ist Puffer- kleiner Recordzeiger ?
DFF3  20 3C E0  JSR $E03C      nein, Record in Puffer schreiben
DFF6  A6 82     LDX $82        Nummer des aktuellen Kanals
DFF8  B5 C1     LDA $C1,X      dazugehörigen Zeiger auf Record holen
DFFA  20 C8 D4  JSR $D4C8      und Pufferzeiger entsprechend setzen
DFFD  A1 99     LDA ($99,X)    Datenbyte aus Puffer holen
DFFF  85 85     STA $85        und merken
E001  A9 20     LDA #$20       Flag für 'Record voll'
E003  20 9D DD  JSR $DD9D      löschen
E006  20 04 E3  JSR $E304      Recordlänge zu Pufferzeiger addieren
E009  48        PHA            neuen Zeigerwert merken
E00A  90 28     BCC $E034      Passt Record noch in aktuellen Sektor ?
E00C  A9 00     LDA #$00       nein, Positionszeiger setzen
```

```
E00E  20 F6 D4   JSR $D4F6      und Byte (Spurnummer) aus Puffer holen
E011  D0 21      BNE $E034      ist ein weiterer Datenblock vorhanden ?
E013  68         PLA            nein, neuen Pufferzeiger wieder holen
E014  C9 02      CMP #$02       und mit Wert für Datenstart vergleichen
E016  F0 12      BEQ $E02A      ist der neue Puffer leer ?
E018  A9 80      LDA #$80       nein, Flag für letztes Byte (EOI)
E01A  20 97 DD   JSR $DD97      setzen
E01D  20 2F D1   JSR $D12F      Puffer- und Kanalnummer feststellen
E020  B5 99      LDA $99,X      Low-Byte des Pufferzeigers holen und
E022  99 44 02   STA $0244,Y    als letztes Zeichen merken
E025  A9 0D      LDA #$0D       'Return' an Ausgabe
E027  85 85      STA $85        übergeben
E029  60         RTS            zurück zur aufrufenden Routine
E02A  20 35 E0   JSR $E035      Zeiger auf letztes Zeichen setzen
E02D  A6 82      LDX $82        Nummer des aktuellen Kanals
E02F  A9 00      LDA #$00       Zeiger auf nächsten Record
E031  95 C1      STA $C1,X      löschen
E033  60         RTS            zurück zur aufrufenden Routine
-----------------------------------------------------------------

[E00A/E011]
Zeiger auf letztes Zeichen setzen
E034  68         PLA            Zeiger auf Start des nächsten Records
E035  A6 82      LDX $82        Nummer des aktuellen Kanals holen und
E037  95 C1      STA $C1,X      Zeiger merken
E039  4C 6E E1   JMP $E16E      Zeiger auf letztes Zeichen setzen
-----------------------------------------------------------------

[DFF3/E0A7/E135]
Sektor des Records bereitstellen
E03C  20 D3 D1   JSR $D1D3      angewähltes Laufwerk feststellen
E03F  20 95 DE   JSR $DE95      Spur und Sektor des nächsten Blocks
E042  20 9E DF   JSR $DF9E      Pufferstatus prüfen
E045  50 16      BVC $E05D      wurde Pufferinhalt geändert ?
E047  20 5E DE   JSR $DE5E      ja, Puffer auf Diskette schreiben
E04A  20 1E CF   JSR $CF1E      neuen Puffer einrichten
E04D  A9 02      LDA #$02       Pufferzeiger auf Beginn des
E04F  20 C8 D4   JSR $D4C8      Datenbereichs setzen
E052  20 AB DD   JSR $DDAB      letzten Job auf Schreiben prüfen
E055  D0 24      BNE $E07B      wurde vorher Sektor geschrieben ?
E057  20 57 DE   JSR $DE57      ja, Sektor wieder in Puffer lesen
E05A  4C 99 D5   JMP $D599      und warten bis Job ausgeführt ist
E05D  20 1E CF   JSR $CF1E      neuen Puffer einrichten
```

```
E060     20 AB DD   JSR $DDAB      letzten Job auf Schreiben prüfen
E063     D0 06      BNE $E06B      wurde vorher Sektor geschrieben ?
E065     20 57 DE   JSR $DE57      ja, Sektor von Diskette einlesen
E068     20 99 D5   JSR $D599      und warten bis Job ausgeführt ist
E06B[1]  20 95 DE   JSR $DE95      Spur und Sektor des nächsten Blocks
E06E     A5 80      LDA $80        Nummer der nächsten Spur holen
E070     F0 09      BEQ $E07B      ist ein weiterer Sektor vorhanden ?
E072     20 1E CF   JSR $CF1E      ja, Puffer neu anlegen
E075     20 57 DE   JSR $DE57      Sektor von Diskette einlesen
E078     20 1E CF   JSR $CF1E      und neuen Puffer anlegen
E07B[2]  60         RTS            zurück zur aufrufenden Routine
-----------------------------------------------------------------------
[E0B4/E0FE]
ein Zeichen eines Records in Puffer schreiben
E07C     20 05 E1   JSR $E105      Flag für 'Puffer geändert' setzen
E07F     20 93 DF   JSR $DF93      Nummer des aktuellen Puffers holen
E082     0A         ASL A          und verdoppeln (da Pufferzeigertabelle
E083     AA         TAX            aus 2-Byte-Werten besteht)
E084     A5 85      LDA $85        einzutragendes Byte holen
E086     81 99      STA ($99,X)    und in Puffer schreiben
E088     B4 99      LDY $99,X      Pufferzeiger (Low-Byte) holen
E08A     C8         INY            und auf nächste Position setzen
E08B     D0 09      BNE $E096      Ende des Puffers erreicht ?
E08D     A4 82      LDY $82        ja, derzeitige Kanalnummer
E08F     B9 C1 00   LDA $00C1,Y    und Zeiger auf den nächsten Record holen
E092     F0 0A      BEQ $E09E      Zeiger gesetzt ?
E094     A0 02      LDY #$02       ja, Pufferzeiger auf
E096[1]  98         TYA            Beginn des Datenbereichs setzen
E097     A4 82      LDY $82        Nummer des aktuellen Kanals holen
E099     D9 C1 00   CMP $00C1,Y    Puffer- und Recordzeiger vergleichen
E09C     D0 05      BNE $E0A3      steht Recordzeiger auf Pufferanfang ?
E09E[1]  A9 20      LDA #$20       ja, Flag für 'Record voll'
E0A0     4C 97 DD   JMP $DD97      setzen
E0A3[1]  F6 99      INC $99,X      Pufferzeiger auf nächstes Byte richten
E0A5     D0 03      BNE $E0AA      Ende des Puffers erreicht ?
E0A7     20 3C E0   JSR $E03C      ja, Sektor auf Diskette schreiben
E0AA[1]  60         RTS            zurück zur aufrufenden Routine
-----------------------------------------------------------------------
```

```
[CFCB]
Record in Datenpuffer schreiben
E0AB    A9 A0     LDA #$A0      Flags für 'letztes Byte' (EOI) und für
E0AD    20 A6 DD  JSR $DDA6     'Record voll' prüfen
E0B0    D0 27     BNE $E0D9     ein Flag gesetzt ?
E0B2[1] A5 85     LDA $85       nein, Byte aus Eingaberegister holen
E0B4    20 7C E0  JSR $E07C     und in Record schreiben
E0B7    A5 F8     LDA $F8       Flag für 'letztes Byte' (EOI) prüfen
E0B9    F0 0D     BEQ $E0C8     war das das letzte Byte ?
E0BB    60        RTS           ja, zurück zur aufrufenden Routine
E0BC[1] A9 20     LDA #$20      Flag für 'Record voll'
E0BE    20 A6 DD  JSR $DDA6     überprüfen
E0C1    F0 05     BEQ $E0C8     ist der Record schon vollgeschrieben ?
E0C3    A9 51     LDA #$51      ja, Fehlerflag für
E0C5    8D 6C 02  STA $026C     '51 Overflow In Record' setzen
E0C8[2] 20 F3 E0  JSR $E0F3     Rest des Records mit Nullen füllen
E0CB    20 53 E1  JSR $E153     nächsten Record holen
E0CE    AD 6C 02  LDA $026C     Flag für Fehler prüfen
E0D1    F0 03     BEQ $E0D6     ist ein Fehler aufgetreten ?
E0D3    4C C8 C1  JMP $C1C8     ja, Fehlermeldung ausgeben
E0D6[1] 4C BC E6  JMP $E6BC     'Ok' Meldung bereitstellen
E0D9[1] 29 80     AND #$80      Flag für 'letztes Byte' (EOI) prüfen
E0DB    D0 05     BNE $E0E2     ist Flag gesetzt ?
E0DD    A5 F8     LDA $F8       nein, EOI vom seriellen Bus prüfen
E0DF    F0 DB     BEQ $E0BC     ist Flag gesetzt ?
E0E1    60        RTS           ja, zurück zur aufrufenden Routine
E0E2[1] A5 85     LDA $85       Byte vom Eingaberegister holen und
E0E4    48        PHA           merken
E0E5    20 1C E3  JSR $E31C     Relative Datei erweitern
E0E8    68        PLA           Byte wieder zurückholen
E0E9    85 85     STA $85       und merken
E0EB    A9 80     LDA #$80      Flag für 'letztes Byte in Datei' (EOI)
E0ED    20 9D DD  JSR $DD9D     löschen
E0F0    4C B2 E0  JMP $E0B2     Record weiter in Puffer schreiben
-----------------------------------------------------------------------
[E0C8/E101]
restlichen Record mit Leerbytes auffüllen
E0F3    A9 20     LDA #$20      Flag für 'Record voll'
E0F5    20 A6 DD  JSR $DDA6     prüfen
E0F8    D0 0A     BNE $E104     ist gesamter Record gefüllt
E0FA    A9 00     LDA #$00      Wert der Leerbytes
```

```
E0FC  85 85     STA $85        setzen
E0FE  20 7C E0  JSR $E07C      Byte in Record schreiben
E101  4C F3 E0  JMP $E0F3      nächstes Byte auffüllen
E104  60        RTS            zurück zur aufrufenden Routine
------------------------------------------------------------------
[DE09/E07C]
Flag für 'Pufferdaten geändert' setzen
E105  A9 40     LDA #$40       Flag für 'Sektor geändert'
E107  20 97 DD  JSR $DD97      setzen
E10A  20 9E DF  JSR $DF9E      Pufferstatus holen
E10D  09 40     ORA #$40       Flag für 'Puffer geändert'
E10F  AE 57 02  LDX $0257      Nummer des Kanals +7 (zeigt auf $AE)
E112  95 A7     STA $A7,X      Pufferstatus neu setzen
E114  60        RTS            zurück zur aufrufenden Routine
------------------------------------------------------------------
[DE8B]
Flag für 'Pufferdaten geändert' löschen
E115  20 9E DF  JSR $DF9E      Pufferstatus holen
E118  29 BF     AND #$BF       und Flag ausblenden
E11A  AE 57 02  LDX $0257      Nummer des Kanals für 2. Puffer
E11D  95 A7     STA $A7,X      Pufferstatus wieder in Tabelle setzen
E11F  60        RTS            zurück zur aufrufenden Routine
------------------------------------------------------------------
[D3B1/E138:E294]
Byte aus Record holen
E120  A9 80     LDA #$80       Flag für 'letztes Byte' (EOI)
E122  20 A6 DD  JSR $DDA6      überprüfen
E125  D0 37     BNE $E15E      ist das das letzte Byte des Records ?
E127  20 2F D1  JSR $D12F      nein, Pufferzeiger initialisieren
E12A  B5 99     LDA $99,X      Pufferzeiger holen
E12C  D9 44 02  CMP $0244,Y    und auf Endposition des Records prüfen
E12F  F0 22     BEQ $E153      ist das Ende des Records erreicht ?
E131  F6 99     INC $99,X      nein, Pufferzeiger auf nächstes Byte
E133  D0 06     BNE $E13B      ist der Datenpuffer voll ?
E135  20 3C E0  JSR $E03C      ja, Sektor schreiben und Nächsten holen
E138  20 2F D1  JSR $D12F      Pufferzeiger initialisieren
E13B  A1 99     LDA ($99,X)    Byte aus Datenpuffer holen
E13D  99 3E 02  STA $023E,Y    und merken
E140  A9 89     LDA #$89       Flag für Lesen/Schreiben/EOI
E142  99 F2 00  STA $00F2,Y    in Kanalstatus setzen
E145  B5 99     LDA $99,X      Low-Byte des Pufferzeigers holen und mit
```

```
E147  D9 44 02  CMP $0244,Y     Wert für Ende des Records vergleichen
E14A  F0 01     BEQ $E14D       schon ganzer Record gelesen ?
E14C  60        RTS             nein, zurück zur aufrufenden Routine
E14D  A9 81     LDA #$81        Flag für Lesen/Schreiben
E14F  99 F2 00  STA $00F2,Y     in Kanalstatus setzen
E152  60        RTS             zurück zur aufrufenden Routine
-------------------------------------------------------------------
[DCA3/E0CB/E12F]
Record holen und ausgeben
E153  20 D0 DF  JSR $DFD0       nächsten Record holen
E156  20 2F D1  JSR $D12F       Puffer- und Kanalnummer feststellen
E159  A5 85     LDA $85         Byte holen und für Ausgabe
E15B  4C 3D E1  JMP $E13D       bereitstellen
-------------------------------------------------------------------
[E125/E262/E26F]
Fehler aufgetreten
E15E  A6 82     LDX $82         Nummer des aktuellen Kanals holen
E160  A9 0D     LDA #$0D        und Ausgabe mit 'Return'
E162  9D 3E 02  STA $023E,X     abschließen
E165  A9 81     LDA #$81        Kanalstatus wieder
E167  95 F2     STA $F2,X       zurücksetzen
E169  A9 50     LDA #$50        Fehlermeldung
E16B  20 C8 C1  JSR $C1C8       '50 Record Not Present' ausgeben
-------------------------------------------------------------------
[E039]
Zeiger auf letztes Zeichen des Records setzen
E16E  A6 82     LDX $82         Nummer des derzeitigen Kanals
E170  B5 C1     LDA $C1,X       Zeiger auf Beginn des nächsten Records
E172  85 87     STA $87         holen und merken
E174  C6 87     DEC $87         Zeiger korregieren (da 0 auch zählt)
E176  C9 02     CMP #$02        und mit Wert für Datenbeginn vergleichen
E178  D0 04     BNE $E17E       steht Zeiger auf Pufferbeginn ?
E17A  A9 FF     LDA #$FF        ja, Zeiger auf Ende des Puffers setzen
E17C  85 87     STA $87         und merken
E17E  B5 C7     LDA $C7,X       Länge des Records holen
E180  85 88     STA $88         und merken
E182  20 E8 D4  JSR $D4E8       aktuellen Pufferzeiger setzen
E185  A6 82     LDX $82         Nummer des derzeitigen Kanals holen
E187  C5 87     CMP $87         Puffer- mit Recordzeiger vergleichen
E189  90 19     BCC $E1A4       ist der Pufferzeiger größer ?
E18B  F0 17     BEQ $E1A4       ja, sind beide Zeiger gleich ?
```

```
E18D  20 1E CF   JSR $CF1E      nein, neuen Puffer anlegen
E190  20 B2 E1   JSR $E1B2      Ende des Records suchen
E193  90 08      BCC $E19D      gefunden ?
E195  A6 82      LDX $82        nein, aktuelle Kanalnummer holen
E197  9D 44 02   STA $0244,X    und Zeiger merken
E19A  4C 1E CF   JMP $CF1E      neuen Puffer anlegen und beenden
E19D[1] 20 1E CF JSR $CF1E      neuen Puffer anlegen
E1A0  A9 FF      LDA #$FF       Recordzeiger auf Ende des Puffers
E1A2  85 87      STA $87        setzen
E1A4[2] 20 B2 E1 JSR $E1B2      Ende des Records suchen
E1A7  B0 03      BCS $E1AC      Ende gefunden ?
E1A9  20 E8 D4   JSR $D4E8      ja, aktuellen Pufferzeiger setzen
E1AC[1] A6 82    LDX $82        Nummer des dazugehörigen Kanals holen
E1AE  9D 44 02   STA $0244,X    und Endposition des Records merken
E1B1  60         RTS            zurück zur aufrufenden Routine
--------------------------------------------------------------------
[E190/E1A4]
Ende des Records suchen
E1B2  20 2B DE   JSR $DE2B      Zeiger auf Pufferbeginn setzen
E1B5  A4 87      LDY $87        aktuellen Recorddzeiger holen
E1B7[1] B1 94    LDA ($94),Y    Byte aus Record lesen
E1B9  D0 0D      BNE $E1C8      Byte gleich Leerbyte ?
E1BB  88         DEY            ja, Pufferzeiger auf nächstes Byte und
E1BC  C0 02      CPY #$02       mit Wert für Pufferbeginn vergleichen
E1BE  90 04      BCC $E1C4      Pufferanfang erreicht ?
E1C0  C6 88      DEC $88        nein, Recordlänge vermindern
E1C2  D0 F3      BNE $E1B7      schon ganzen Recordbereich abgesucht ?
E1C4[1] C6 88    DEC $88        ja, Recordzeiger korregieren
E1C6  18         CLC            Flag für 'Ende gefunden' setzen
E1C7  60         RTS            zurück zur aufrufenden Routine
E1C8[1] 98       TYA            aktuelle Pufferposition holen und
E1C9  38         SEC            Flag für 'Ende nicht gefunden' setzen
E1CA  60         RTS            zurück zur aufrufenden Routine
--------------------------------------------------------------------
[CA56/CA69/DB32/DF0B/E31F]
Ende der Relativen Datei suchen
E1CB  20 D2 DE   JSR $DED2      Nummer des aktuellen Side-Sektors holen
E1CE  85 D5      STA $D5        und merken
E1D0  A9 04      LDA #$04       Pufferzeiger auf Beginn
E1D2  85 94      STA $94        des Sektors zurücksetzen
E1D4  A0 0A      LDY #$0A       Zeiger auf Spur des letzten Side-Sektors
```

```
E1D6     D0 04      BNE $E1DC       immer Sprung nach $E1DC
E1D8[1]  88         DEY             Pufferzeiger auf Spurnummer
E1D9     88         DEY             des vorangehenden Side-Sektors setzen
E1DA     30 26      BMI $E202       kein Side-Sektor mehr vorhanden ?
E1DC[1]  B1 94      LDA ($94),Y     nein, Spurnummer des Side-Sektors holen
E1DE     F0 F8      BEQ $E1D8       ist Sektor angelegt ?
E1E0     98         TYA             ja, Nummer des Side-Sektors
E1E1     4A         LSR A           ermitteln
E1E2     C5 D5      CMP $D5         und mit aktueller Nummer vergleichen
E1E4     F0 09      BEQ $E1EF       identisch ?
E1E6     85 D5      STA $D5         nein, neue Side-Sektornummer merken
E1E8     A6 82      LDX $82         Nummer des derzeitigen Kanals holen
E1EA     B5 CD      LDA $CD,X       Puffer des Sektors feststellen
E1EC     20 1B DF   JSR $DF1B       und Sektor einlesen
E1EF[1]  A0 00      LDY #$00        Pufferzeiger auf Beginn des
E1F1     84 94      STY $94         Sektors zurücksetzen
E1F3     B1 94      LDA ($94),Y     Spur des nächsten Sektors holen
E1F5     D0 0B      BNE $E202       kein weiterer Side-Sektor ?
E1F7     C8         INY             ja, Zeiger auf nächste Position setzen
E1F8     B1 94      LDA ($94),Y     Zahl der gültigen Datenbytes holen
E1FA     A8         TAY             und merken
E1FB     88         DEY             Zeiger auf Verkettungsbytes des Sektors
E1FC     84 D6      STY $D6         des letzten Records
E1FE     98         TYA             setzen und merken
E1FF     4C E9 DE   JMP $DEE9       Pufferzeiger auf diese Position setzen
E202[1]  A9 67      LDA #$67        Fehlermeldung
E204     20 45 E6   JSR $E645       '67 Illegal Track Or Sector' ausgeben
-----------------------------------------------------------------------------
[Einsprung über Routine C146]
Routine für Record-Befehl ('P')
E207     20 B3 C2   JSR $C2B3       Zeiger für Befehlsstring setzen
E20A     AD 01 02   LDA $0201       zweites Zeichen des Befehls aus Puffer
E20D     85 83      STA $83         holen und als Sekundäradresse festlegen
E20F     20 EB D0   JSR $D0EB       Lesekanal öffnen
E212     90 05      BCC $E219       wurde ein freier Kanal gefunden ?
E214     A9 70      LDA #$70        nein, Fehlermeldung
E216     20 C8 C1   JSR $C1C8       '70 No Channel' ausgeben
E219[1]  A9 A0      LDA #$A0        Flags für EOI
E21B     20 9D DD   JSR $DD9D       löschen
E21E     20 25 D1   JSR $D125       Dateityp holen und prüfen
E221     F0 05      BEQ $E228       liegt eine Relative Datei vor ?
```

```
E223      A9 64     LDA #$64     nein, Fehlermeldung
E225      20 C8 C1  JSR $C1C8    '64 File Type Mismatch' ausgeben
E228[1]   B5 EC     LDA $EC,X    Flags des Kanals holen und
E22A      29 01     AND #$01     angewählltes Laufwerk
E22C      85 7F     STA $7F      als aktuelles Laufwerk übernehmen
E22E      AD 02 02  LDA $0202    drittes Zeichen aus Eingabepuffer holen
E231      95 B5     STA $B5,X    und als Low-Byte der Recordnummer setzen
E233      AD 03 02  LDA $0203    High-Byte der Recordnummer holen
E236      95 BB     STA $BB,X    und übernehmen
E238      A6 82     LDX $82      Nummer des derzeitigen Kanals holen
E23A      A9 89     LDA #$89     Flag für Lesen/Schreiben/EOI
E23C      95 F2     STA $F2,X    in Kanalstatus setzen
E23E      AD 04 02  LDA $0204    fünftes Zeichen aus Eingabepuffer holen
E241      F0 10     BEQ $E253    keine Angabe ?
E243      38        SEC          nein, Position in Record übernehmen
E244      E9 01     SBC #$01     und prüfen ob Zeiger auf 1
E246      F0 0B     BEQ $E253    Zeiger auf Anfang des Records gesetzt ?
E248      D5 C7     CMP $C7,X    nein, mit Länge des Records vergleichen
E24A      90 07     BCC $E253    ist Position zulässig ?
E24C      A9 51     LDA #$51     nein, Fehler '51 Overflow In Record'
E24E      8D 6C 02  STA $026C    in Fehlerflag speichern
E251      A9 00     LDA #$00     Positionszeiger auf Beginn des Records
E253[3]   85 D4     STA $D4      setzen
E255      20 0E CE  JSR $CE0E    Position des Records errechnen
E258      20 F8 DE  JSR $DEF8    dazugehörigen Side-Sektor einlesen
E25B      50 08     BVC $E265    Side-Sektor fehlerfrei gelesen ?
E25D      A9 80     LDA #$80     nein, Flag für 'letztes Byte' (EOI)
E25F      20 97 DD  JSR $DD97    setzen
E262      4C 5E E1  JMP $E15E    Fehler '50 Record Not Present' ausgeben
E265[1]   20 75 E2  JSR $E275    gesuchten Record lesen
E268      A9 80     LDA #$80     Flag für 'letztes Byte'
E26A      20 A6 DD  JSR $DDA6    testen
E26D      F0 03     BEQ $E272    ist Record nicht vorhanden ?
E26F      4C 5E E1  JMP $E15E    ja, Fehler '50 Record Not Present'
E272[1]   4C 94 C1  JMP $C194    'Ok' Meldung bereitstellen
----------------------------------------------------------------------

[E265/E441]
gesuchten Record in Puffer einlesen
E275      20 9C E2  JSR $E29C    Sektor, der Record enthält, einlesen
E278      A5 D7     LDA $D7      Position im Record in
E27A      20 C8 D4  JSR $D4C8    aktuellen Pufferzeiger übertragen
```

```
E27D  A6 82     LDX $82        Nummer des derzeitigen Kanals
E27F  B5 C7     LDA $C7,X      Länge des Records feststellen
E281  38        SEC            und davon aktuelle Position
E282  E5 D4     SBC $D4        in Datensatz subtrahieren
E284  B0 03     BCS $E289      ist Zeiger noch im Datensatz ?
E286  4C 02 E2  JMP $E202      ja, Fehler '67 Illegal Track Or Sector'
E289  18        CLC            Position des gewünschten Bytes
E28A  65 D7     ADC $D7        im Record einrechnen
E28C  90 03     BCC $E291      Byte im nächsten Datensektor ?
E28E  69 01     ADC #$01       ja, Position auf Start setzen
E290  38        SEC            und Flag für nächsten Sektor setzen
E291  20 09 E0  JSR $E009      Zeiger für auf Record setzen
E294  4C 38 E1  JMP $E138      Byte aus Record holen
E297  A9 51     LDA #$51       Fehlermeldung
E299  20 C8 C1  JSR $C1C8      '51 Overflow In Record' ausgeben
-----------------------------------------------------------------------
[CA6C/E322/E275]
Sektor der Record enthält in Puffer einlesen
E29C  A5 94     LDA $94        aktuellen Pufferzeiger
E29E  85 89     STA $89        in Zwischenspeicher
E2A0  A5 95     LDA $95        in Adresse $89/$8A
E2A2  85 8A     STA $8A        retten
E2A4  20 D0 E2  JSR $E2D0      Puffer auf Sektor prüfen
E2A7  D0 01     BNE $E2AA      ist der gesuchte Sektor im Puffer ?
E2A9  60        RTS            ja, zurück zur aufrufenden Routine
E2AA  20 F1 DD  JSR $DDF1      Pufferinhalt auf Diskette schreiben
E2AD  20 0C DE  JSR $DE0C      Spur und Sektor des nächsten Blocks
E2B0  A5 80     LDA $80        Spurnummer des nächsten Sektors holen
E2B2  F0 0E     BEQ $E2C2      ist weiterer Sektor vorhanden ?
E2B4  20 D3 E2  JSR $E2D3      ja, Puffer auf Sektor prüfen
E2B7  D0 06     BNE $E2BF      ist Sektor schon im Puffer ?
E2B9  20 1E CF  JSR $CF1E      ja, neuen Puffer anlegen
E2BC  4C DA D2  JMP $D2DA      alle inaktiven Puffer freigeben
E2BF  20 DA D2  JSR $D2DA      alle inaktiven Puffer freigeben
E2C2  A0 00     LDY #$00       Pufferzeiger initialisieren
E2C4  B1 89     LDA ($89),Y    Spurnummer aus Side-Sektor holen
E2C6  85 80     STA $80        und als aktuelle Spur übernehmen
E2C8  C8        INY            Pufferzeiger auf nächstes Byte
E2C9  B1 89     LDA ($89),Y    Nummer des Datensektors holen
E2CB  85 81     STA $81        und speichern
E2CD  4C AF D0  JMP $D0AF      Sektor in Puffer einlesen
```

```
--------------------------------------------------------------------
[E2A4]
prüfen, ob Sektor schon im Puffer ist
E2D0  20 3E DE   JSR $DE3E      Spur und Sektor des letzten Jobs holen
E2D3  A0 00      LDY #$00       Pufferzeiger initialisieren
E2D5  B1 89      LDA ($89),Y    gesuchte Spur aus Side-Sektor holen und
E2D7  C5 80      CMP $80        mit zuletzt gelesenem Wert vergleichen
E2D9  F0 01      BEQ $E2DC      identisch ?
E2DB  60         RTS            nein, zurück zur aufrufenden Routine
E2DC  C8         INY            Pufferzeiger auf Sektornummer setzen
E2DD  B1 89      LDA ($89),Y    Nummer des gesuchten Sektors holen und
E2DF  C5 81      CMP $81        mit aktuellem Sektor vergleichen
E2E1  60         RTS            zurück zur aufrufenden Routine
--------------------------------------------------------------------
[DD7B/E3C2/E3CE]
Neue Records in Sektor anlegen
E2E2  20 2B DE   JSR $DE2B      aktuelle Pufferadresse setzen
E2E5  A0 02      LDY #$02       Zeiger auf Beginn des Datenbereichs
E2E7  A9 00      LDA #$00       Wert zum Löschen des Sektors
E2E9  91 94      STA ($94),Y    Leerbyte in Puffer schreiben
E2EB  C8         INY            Pufferzeiger auf nächstes Byte setzen
E2EC  D0 FB      BNE $E2E9      schon ganzer Puffer gefüllt ?
E2EE  20 04 E3   JSR $E304      ja, Position des nächsten Records holen
E2F1  95 C1      STA $C1,X      und merken
E2F3  A8         TAY            Wert als Pufferzeiger übernehmen
E2F4  A9 FF      LDA #$FF       Wert zum Anlegen des Records
E2F6  91 94      STA ($94),Y    in Puffer schreiben
E2F8  20 04 E3   JSR $E304      Position des nächsten Records errechnen
E2FB  90 F4      BCC $E2F1      hat der Record im Sektor noch Platz ?
E2FD  D0 04      BNE $E303      nein, passt Record genau in Sektor ?
E2FF  A9 00      LDA #$00       ja, Position des nächsten Records
E301  95 C1      STA $C1,X      auf Anfang des nächsten Sektors setzen
E303  60         RTS            zurück zur aufrufenden Routine
--------------------------------------------------------------------
[E006/E2EE/E2F8]
Berechnet die Position des neuen Records im Sektor
E304  A6 82      LDX $82        Nummer des derzeitigen Kanals holen
E306  B5 C1      LDA $C1,X      und dazugehörigen Recordzeiger holen
E308  38         SEC            Flag für 'keinen weitern Record' setzen
E309  F0 0D      BEQ $E318      füllt alter Record den Sektor genau ?
E30B  18         CLC            nein, Länge des
```

```
E30C  75 C7     ADC $C7,X    Record zur aktuellen Position addieren
E30E  90 0B     BCC $E31B    reicht Record noch in nächsten Sektor ?
E310  D0 06     BNE $E318    ja, füllt Record den Sektor genau ?
E312  A9 02     LDA #$02     ja, Zeiger auf Start des neuen Sektors
E314  2C CC FE  BIT $FECC    Flags für 'noch ein Sektor' setzen
E317  60        RTS          zurück zur aufrufenden Routine
E318  69 01     ADC #$01     Zeiger auf Beginn des nächten Records
E31A  38        SEC          Flag für 'kein weiterer Sektor' setzen
E31B  60        RTS          zurück zur aufrufenden Routine
-----------------------------------------------------------------
[E0E5/E33B:CA85]
Neue Records an Relative Datei anfügen
E31C  20 D3 D1  JSR $D1D3    Nummer des angewählten Laufwerks holen
E31F  20 CB E1  JSR $E1CB    Position des letzten Records holen
E322  20 9C E2  JSR $E29C    Side-Sektoren und Records einlesen
E325  20 7B CF  JSR $CF7B    neuen Puffer anlegen
E328  A5 D6     LDA $D6      Zeiger auf Datenblock in Side-Sektor
E32A  85 87     STA $87      retten
E32C  A5 D5     LDA $D5      Zeiger auf aktuellen Side-Sektor
E32E  85 86     STA $86      zwischenspeichern
E330  A9 00     LDA #$00     Flag für 'nur ein Block'
E332  85 88     STA $88      löschen
E334  A9 00     LDA #$00     Zeiger auf Position des
E336  85 D4     STA $D4      Records löschen
E338  20 0E CE  JSR $CE0E    Side-Sektor des Datenblocks errechnen
E33B  20 4D EF  JSR $EF4D    Zahl der freien Blöcke holen
E33E  A4 82     LDY $82      Nummer des aktuellen Kanals feststellen
E340  B6 C7     LDX $C7,Y    Länge des zugehörigen Records holen
E342  CA        DEX          korregieren
E343  8A        TXA          (da 0 auch gezählt wird)
E344  18        CLC          und zu aktuellem
E345  65 D7     ADC $D7      Pufferzeiger addieren
E347  90 0C     BCC $E355    liegt neuer Pufferzeiger im Sektor ?
E349  E6 D6     INC $D6      nein, Zeiger in Side-Sektor auf Spur-
E34B  E6 D6     INC $D6      und Sektor des nächsten Datenblocks
E34D  D0 06     BNE $E355    Zeiger noch im aktuellen Side-Sektor ?
E34F  E6 D5     INC $D5      nein, nächsten Side-Sektor anwählen
E351  A9 10     LDA #$10     Pufferzeiger auf Beginn der Spur- und
E353  85 D6     STA $D6      Sektorzeiger der Datenblöcke
E355  A5 87     LDA $87      alten Pufferzeiger auf
E357  18        CLC          nächste Sektordaten (Spur/Sektor)
```

```
E358     69 02     ADC #$02     setzen
E35A     20 E9 DE  JSR $DEE9    Pufferzeiger setzen
E35D     A5 D5     LDA $D5      Nummer des aktuellen Side-Sektors holen
E35F     C9 06     CMP #$06     und mit maximalem Wert vergleichen
E361     90 05     BCC $E368    Nummer zulässig ?
E363[2]  A9 52     LDA #$52     nein, Fehlermeldung
E365     20 C8 C1  JSR $C1C8    '52 File Too Large' ausgeben
E368[1]  A5 D6     LDA $D6      aktuelle Position im Side-Sektor
E36A     38        SEC          holen
E36B     E5 87     SBC $87      letzten Side-Sektor-Zeiger subtrahieren
E36D     B0 03     BCS $E372    ist neuer Wert in vorhergehndem Sektor ?
E36F     E9 0F     SBC #$0F     ja, Verkettungsbytes am Anfang des
E371     18        CLC          Side-Sektors beachten
E372[1]  85 72     STA $72      und neuen Wert merken
E374     A5 D5     LDA $D5      aktuelle Side-Sektornummer holen
E376     E5 86     SBC $86      und letzte Nummer abziehen
E378     85 73     STA $73      neuen Wert merken
E37A     A2 00     LDX #$00     Zwischenspeicher
E37C     86 70     STX $70      für Zahl der benutzten
E37E     86 71     STX $71      Blöcke löschen
E380     AA        TAX          Side-Sektornummer 0
E381     20 51 DF  JSR $DF51    Zahl der von der Datei benötigten Blöcke
E384     A5 71     LDA $71      errechnen und holen (High-Byte)
E386     D0 07     BNE $E38F    ist Zahl der Blöcke kleiner 256 ?
E388     A6 70     LDX $70      ja, Low-Byte der Blockzahl
E38A     CA        DEX          prüfen
E38B     D0 02     BNE $E38F    ist nur ein Block (Side-Sektor) belegt ?
E38D     E6 88     INC $88      Flag für 'nur ein Block' setzen
E38F[2]  CD 73 02  CMP $0273    mit Zahl der freien Blöcke vergleichen
E392     90 09     BCC $E39D    ist auf der Diskette noch Platz frei ?
E394     D0 CD     BNE $E363    nein, paßt Datei evtl. genau auf Disk ?
E396     AD 72 02  LDA $0272    ja, Low-Bytes der benötigten Blocks mit
E399     C5 70     CMP $70      Zahl der freien Blocks vergleichen
E39B     90 C6     BCC $E363    ist Datei größer als Kapazität ?
E39D[1]  A9 01     LDA #$01     nein, Pufferzeiger auf Sektornummer
E39F     20 F6 D4  JSR $D4F6    Byte aus Puffer holen
E3A2     18        CLC          Zeiger auf aktuelles
E3A3     69 01     ADC #$01     Datenbyte in aktuellem Sektor erhöhen
E3A5     A6 82     LDX $82      Kanalnummer holen
E3A7     95 C1     STA $C1,X    Zeiger auf Datenbyte merken
E3A9     20 1E F1  JSR $F11E    nächsten freien Sektor aus BAM holen
```

```
E3AC    20 FD DD   JSR $DDFD    Verkettungsbytes für nächsten Sektor
E3AF    A5 88      LDA $88      Flag für 'nur ein Block'
E3B1    D0 15      BNE $E3C8    gesetzt ?
E3B3    20 5E DE   JSR $DE5E    nein, Sektor auf Diskette schreiben
E3B6[2] 20 1E CF   JSR $CF1E    Puffer wechseln
E3B9    20 D0 D6   JSR $D6D0    Spur und Sektor an Jobschleife
E3BC    20 1E F1   JSR $F11E    nächsten freien Block in BAM suchen
E3BF    20 FD DD   JSR $DDFD    Parameter des nächsten Blocks in Puffer
E3C2    20 E2 E2   JSR $E2E2    neue Records anlegen
E3C5    4C D4 E3   JMP $E3D4    Sektor auf Diskette schreiben
E3C8[2] 20 1E CF   JSR $CF1E    Puffer wechseln
E3CB    20 D0 D6   JSR $D6D0    Spur und Sektor an Jobschleife
E3CE    20 E2 E2   JSR $E2E2    neue Records anlegen
E3D1    20 19 DE   JSR $DE19    letzten Sektor kennzeichnen
E3D4[1] 20 5E DE   JSR $DE5E    Sektor auf Diskette schreiben
E3D7    20 0C DE   JSR $DE0C    Spur und Sektor von Verkettungsbytes
E3DA    A5 80      LDA $80      nächste Spurnummer holen und
E3DC    48         PHA          merken
E3DD    A5 81      LDA $81      nächste Sektornummer
E3DF    48         PHA          retten
E3E0    20 3E DE   JSR $DE3E    Spur und Sektor des letzten Jobs holen
E3E3    A5 81      LDA $81      Nummer des letzten Sektors
E3E5    48         PHA          merken
E3E6    A5 80      LDA $80      Spurnummer des letzten Sektors
E3E8    48         PHA          retten
E3E9    20 45 DF   JSR $DF45    Pufferzeiger für Side-Sektor setzen
E3EC    AA         TAX          und Low-Byte merken
E3ED    D0 0A      BNE $E3F9    steht Zeiger auf Pufferstart ?
E3EF    20 4E E4   JSR $E44E    ja, neuen Side-Sektor anlegen
E3F2    A9 10      LDA #$10     Pufferzeiger auf Beginn der Zeiger
E3F4    20 E9 DE   JSR $DEE9    auf Datensektoren setzen
E3F7    E6 86      INC $86      alte Side-Sektornummer erhöhen
E3F9[1] 68         PLA          Spur des letzten Sektors wieder holen
E3FA    20 8D DD   JSR $DD8D    und in Side-Sektor eintragen
E3FD    68         PLA          Sektornummer holen
E3FE    20 8D DD   JSR $DD8D    und Byte in Side-Sektor übernehmen
E401    68         PLA          Nummer des aktuellen Sektors
E402    85 81      STA $81      holen und speichern
E404    68         PLA          Nummer der aktuellen Spur
E405    85 80      STA $80      holen und speichern
E407    F0 0F      BEQ $E418    letzter Block ?
```

```
E409  A5 86     LDA $86       nein, aktuelle Side-Sektor-Nummer
E40B  C5 D5     CMP $D5       mit der Letzten vergleichen
E40D  D0 A7     BNE $E3B6     hat sie sich geändert ?
E40F  20 45 DF  JSR $DF45     ja, Pufferzeiger positionieren
E412  C5 D6     CMP $D6       und mit Side-Sektor-Zeiger vergleichen
E414  90 A0     BCC $E3B6     ist der Pufferzeiger kleiner ?
E416  F0 B0     BEQ $E3C8     nein, ist er gleich ?
E418[1] 20 45 DF JSR $DF45    nein, Pufferzeiger positionieren
E41B  48        PHA           und als Endmarkierung merken
E41C  A9 00     LDA #$00      Pufferzeiger auf
E41E  20 DC DE  JSR $DEDC     Null zurücksetzen
E421  A9 00     LDA #$00      Pufferzeiger auf Anfang
E423  A8        TAY           des Sektors setzen
E424  91 94     STA ($94),Y   Flag für letzter Block im Puffer setzen
E426  C8        INY           Pufferzeiger auf nächstes Byte setzen
E427  68        PLA           Zeiger auf Ende zurückholen
E428  38        SEC           und um eins
E429  E9 01     SBC #$01      verringern
E42B  91 94     STA ($94),Y   Zeiger für Zahl der gültigen Bytes
E42D  20 6C DE  JSR $DE6C     Sektor auf Diskette schreiben
E430  20 99 D5  JSR $D599     und auf Schreibfehler prüfen
E433  20 F4 EE  JSR $EEF4     Sektor in BAM belegen
E436  20 0E CE  JSR $CE0E     Zeiger für REL-Datei neu initialisieren
E439  20 1E CF  JSR $CF1E     anderen Puffer anwählen
E43C  20 F8 DE  JSR $DEF8     Side-Sektor prüfen
E43F  70 03     BVS $E444     ist richtiger Side-Sektor im Puffer ?
E441  4C 75 E2  JMP $E275     ja, Recordzeiger neu setzen; Ende
E444[1] A9 80   LDA #$80      Zeiger und Flags
E446  20 97 DD  JSR $DD97     des Filetyps zurücksetzen
E449  A9 50     LDA #$50      Fehlermeldung
E44B  20 C8 C1  JSR $C1C8     '50 Record Not Present' ausgeben
-------------------------------------------------------------------------
[E3EF]
Neuen Side-Sektor fertigstellen
E44E  20 1E F1  JSR $F11E     nächsten freien Block feststellen
E451  20 1E CF  JSR $CF1E     Puffer auswählen
E454  20 F1 DD  JSR $DDF1     vorhergehenden Side-Sektor schreiben
E457  20 93 DF  JSR $DF93     Puffernummer holen
E45A  48        PHA           und merken
E45B  20 C1 DE  JSR $DEC1     Datenpuffer löschen
E45E  A6 82     LDX $82       Nummer des Kanals
```

```
E460  B5 CD     LDA $CD,X     Nummer des Puffers für Side-Sektor
E462  A8        TAY           übernehmen
E463  68        PLA           auf Stapel und in
E464  AA        TAX           X/Y-Register merken
E465  A9 10     LDA #$10      16 Bytes des vorhergehenden Side-Sektors
E467  20 A5 DE  JSR $DEA5     in aktuellen Puffer übertragen
E46A  A9 00     LDA #$00      Wert für Pufferzeiger
E46C  20 DC DE  JSR $DEDC     Pufferzeiger zurücksetzen
E46F  A0 02     LDY #$02      Puffer des vorhergehenden Side-Sektors
E471  B1 94     LDA ($94),Y   anwählen und Side-Sektor-Nummer holen
E473  48        PHA           Nummer des letzten Side-sektors merken
E474  A9 00     LDA #$00      Pufferzeiger auf Puffer des neuen
E476  20 C8 D4  JSR $D4C8     Side-Sektors richten und zurücksetzen
E479  68        PLA           Nummer des letzten Side-Sektors holen
E47A  18        CLC           um eins
E47B  69 01     ADC #$01      erhöhen und
E47D  91 94     STA ($94),Y   als neue Nummer speichern
E47F  0A        ASL A         Wert verdoppeln
E480  69 04     ADC #$04      und 4 addieren
E482  85 89     STA $89       Zeiger auf Spur/Sektor setzen
E484  A8        TAY           und merken
E485  38        SEC           dann daraus
E486  E9 02     SBC #$02      Zeiger auf
E488  85 8A     STA $8A       vorherigen Side-Sektor berechnen
E48A  A5 80     LDA $80       Nummer der Spur
E48C  85 87     STA $87       merken
E48E  91 94     STA ($94),Y   den aktuellen Puffer schreiben
E490  C8        INY           Pufferzeiger auf nächstes Byte setzen
E491  A5 81     LDA $81       Nummer des Sektors
E493  85 88     STA $88       speichern und in den
E495  91 94     STA ($94),Y   aktuellen Puffer übertragen
E497  A0 00     LDY #$00      Pufferzeiger auf Beginn
E499  98        TYA           des Sektors richten
E49A  91 94     STA ($94),Y   Flag für letzten Side-Sektor
E49C  C8        INY           Pufferzeiger auf nächstes Byte setzen
E49D  A9 11     LDA #$11      Zahl der gültigen Bytes des Sektors
E49F  91 94     STA ($94),Y   setzen (17)
E4A1  A9 10     LDA #$10      Pufferzeiger auf
E4A3  20 C8 D4  JSR $D4C8     Position 16 einstellen
E4A6  20 50 DE  JSR $DE50     Sektor auf Diskette schreiben
E4A9  20 99 D5  JSR $D599     Rückmeldung des Diskcontrollers abwarten
```

```
E4AC  A6 82     LDX $82        aktuelle Kanalnummer setzen
E4AE  B5 CD     LDA $CD,X      Nummer des Puffers zum Side-Sektor
E4B0  48        PHA            holen und merken
E4B1  20 9E DF  JSR $DF9E      Puffernummer holen
E4B4  A6 82     LDX $82        aktuelle Kanalnummer
E4B6  95 CD     STA $CD,X      und als dritten Puffer merken
E4B8  68        PLA            Puffernummer für Side-Sektor
E4B9  AE 57 02  LDX $0257      Zeiger auf letzten aktiven Datenpuffer
E4BC  95 A7     STA $A7,X      Puffer belegen
E4BE  A9 00     LDA #$00       Pufferzeiger auf Null
E4C0  20 C8 D4  JSR $D4C8      zurücksetzen
E4C3  A0 00     LDY #$00       Pufferzeiger auf Beginn des Sektors
E4C5  A5 80     LDA $80        Nummer der Spur
E4C7  91 94     STA ($94),Y    in Datenpuffer übernehmen
E4C9  C8        INY            Zeiger auf nächstes Zeichen setzen
E4CA  A5 81     LDA $81        Nummer des Sektors
E4CC  91 94     STA ($94),Y    in den Puffer übernehmen
E4CE  4C DE E4  JMP $E4DE      Side-Sektor auf Diskette schreiben
E4D1  20 93 DF  JSR $DF93      aktuelle Puffernummer feststellen
E4D4  A6 82     LDX $82        aktuelle Kanalnummer
E4D6  20 1B DF  JSR $DF1B      nächsten Side-Sektor von Diskette lesen
E4D9  A9 00     LDA #$00       Pufferzeiger auf Null
E4DB  20 C8 D4  JSR $D4C8      zurücksetzen
E4DE  C6 8A     DEC $8A        Side-Sektornummer
E4E0  C6 8A     DEC $8A        korregieren
E4E2  A4 89     LDY $89        Pufferzeiger für Spur-/Sektorposition
E4E4  A5 87     LDA $87        Nummer der Spur
E4E6  91 94     STA ($94),Y    in den Datenpuffer schreiben
E4E8  C8        INY            Pufferzeiger auf nächstes Byte setzen
E4E9  A5 88     LDA $88        Nummer des Sektors holen
E4EB  91 94     STA ($94),Y    und in den Puffer übernehmen
E4ED  20 5E DE  JSR $DE5E      Side-Sektor auf Diskette schreiben
E4F0  20 99 D5  JSR $D599      Meldung des Diskcontrollers abwarten
E4F3  A4 8A     LDY $8A        Side-Sektornummer holen
E4F5  C0 03     CPY #$03       und prüfen
E4F7  B0 D8     BCS $E4D1      größer drei ?
E4F9  4C 1E CF  JMP $CF1E      nein, anderen Puffer anwählen
--------------------------------------------------------------------
```

Das erste Byte ist die Fehlernummer in BCD-Code
Danach folgt der Text der Fehlermeldung. Dabei wird Anfang und Ende des Textstrings durch Bit7=1 im ersten und letzten Byte gekennzeichnet
Einige Wörter sind durch Kurzkodes ersetzt. Die höherwertige Bytehälfte dieser Werte ist Null. Sie werden wie Fehlermeldungen gehandhabt.

```
-----------------------------------------------------------------------
E4FC 00                              'ok'
E4FD A0 4F CB
-----------------------------------------------------------------------
E500 20 21 22 23 24 27               'read error'
E506 D2 45 41 44 89
-----------------------------------------------------------------------
E50B 52                              'file too large'
E50C 83 20 54 4F 4F 20 4C 41 52 47 C5
-----------------------------------------------------------------------
E517 50                              'record not present'
E518 8B 06 20 50 52 45 53 45 4E D4
-----------------------------------------------------------------------
E522 51                              'overflow in record'
E523 CF 56 45 52 46 4C 4F 57 20 49 4E 8B
-----------------------------------------------------------------------
E52F 25 28                           'write error'
E531 8A 89
-----------------------------------------------------------------------
E533 26                              'write protect on'
E534 8A 20 50 52 4F 54 45 43 54 20 4F CE
-----------------------------------------------------------------------
E540 29                              'disk id mismatch'
E541 88 20 49 44 85
-----------------------------------------------------------------------
E546 30 31 32 33 34                  'syntax error'
E54B D3 59 4E 54 41 58 89
-----------------------------------------------------------------------
E552 60                              'write file open'
E553 8A 03 84
-----------------------------------------------------------------------
E556 63                              'file exists'
E557 83 20 45 58 49 53 54 D3
-----------------------------------------------------------------------
E55F 64                              'file type mismatch'
E560 83 20 54 59 50 45 85
```

```
-----------------------------------------------------------------------
E567 65                           'no block'
E568 CE 4F 20 42 4C 4F 43 CB
-----------------------------------------------------------------------
E570 66 67                        'illegal track or sector'
E572 C9 4C 4C 45 47 41 4C 20 54 52 41 43 4B 20 4F 52 20 53 45 43 54 4F D2
-----------------------------------------------------------------------
E589 61                           'file not open'
E58A 83 06 84
-----------------------------------------------------------------------
E58D 39 62                        'file not found'
E58F 83 06 87
-----------------------------------------------------------------------
E592 01                           'files scratched'
E593 83 53 20 53 43 52 41 54 43 48 45 C4
-----------------------------------------------------------------------
E59F 70                           'no channel'
E5A0 CE 4F 20 43 48 41 4E 4E 45 CC
-----------------------------------------------------------------------
E5AA 71                           'dir error'
E5AB C4 49 52 89
-----------------------------------------------------------------------
E5AF 72                           'disk full'
E5B0 88 20 46 55 4C CC
-----------------------------------------------------------------------
E5B6 73                           'cbm dos v3.0 1571'
E5B7 C3 42 4D 20 44 4F 53 20 56 33 2E 30 20 31 35 37 B1
-----------------------------------------------------------------------
E5C8 74                           'drive not ready'
E5C9 C4 52 49 56 45 06 20 52 45 41 44 D9
-----------------------------------------------------------------------
Oft benötigte Wörter und ihre Kurzcodes :
-----------------------------------------------------------------------
E5D5 09                           'error'
E5D6 C5 52 52 4F D2
-----------------------------------------------------------------------
E5DB 0A                           'write'
E5DC D7 52 49 54 C5
-----------------------------------------------------------------------
E5E1 03                           'file'
E5E2 C6 49 4C C5
```

```
--------------------------------------------------------------------
E5E6 04                          'open'
E5E7 CF 50 45 CE
--------------------------------------------------------------------
E5EB 05                          'mismatch'
E5EC CD 49 53 4D 41 54 43 C8
--------------------------------------------------------------------
E5F4 06                          'not'
E5F5 CE 4F D4
--------------------------------------------------------------------
E5F8 07                          'found'
E5F9 C6 4F 55 4E C4
--------------------------------------------------------------------
E5FE 08                          'disk'
E5FF C4 49 53 CB
--------------------------------------------------------------------
E603 0B                          'record'
E604 D2 45 43 4F 52 C4
--------------------------------------------------------------------
[8391/91AA/A47B/A6CB/BF63/C8EC/D641/E60D:A9D2]
Ausgabe der Fehlermeldung (A muß Fehler- und X die Puffernummer enthalten)
E60A  4C B9 A9   JMP $A9B9     1571 Modus berücksichtigen
E60D  8A         TXA           Puffernummer
E60E  0A         ASL A         verdoppeln
E60F  AA         TAX           und als Zeiger auf Diskcontroller setzen
E610  B5 06      LDA $06,X     Nummer der Spur vom Diskcontroler holen
E612  85 80      STA $80       und speichern
E614  B5 07      LDA $07,X     Nummer des Sektors holen
E616  85 81      STA $81       und speichern
E618  68         PLA           Fehlernummer wieder herstellen
E619  29 0F      AND #$0F      ist es die
E61B  F0 08      BEQ $E625     Fehlernummer 15 oder größer ?
E61D  C9 0F      CMP #$0F      ja, ist es es genau Fehler
E61F  D0 06      BNE $E627     Nummer 15 ?
E621  A9 74      LDA #$74      ja, interne Nummer der Fehlermeldung
E623  D0 08      BNE $E62D     immer Sprung nach $E62D
E625  A9 06      LDA #$06      Fehlernummer
E627  09 20      ORA #$20      Lesefehler umrechnen
E629  AA         TAX           und für
E62A  CA         DEX           Fehlertabelle (BCD-Werte)
E62B  CA         DEX           korregieren
```

```
E62C  8A        TXA        Nummer wieder holen
E62D  48        PHA        Fehlernummer retten
E62E  AD 2A 02  LDA $022A  Nummer des ausgeführten Befehls
E631  C9 00     CMP #$00   auf Validate - Befehl prüfen
E633  D0 0F     BNE $E644  identisch ?
E635  A9 FF     LDA #$FF   ja, dann Nummer des
E637  8D 2A 02  STA $022A  Befehls löschen
E63A  68        PLA        und Fehlernummer wieder zurückholen
E63B  20 C7 E6  JSR $E6C7  Fehlermeldung im Puffer erzeugen
E63E  20 42 D0  JSR $D042  Initialize - Befehl ausführen
E641  4C 48 E6  JMP $E648  Fehlermeldung aktivieren
E644  68        PLA        Fehlernummer zurückholen
----------------------------------------------------------------------
[A582/A9F5/CD2E/D54F/D577/DC03/E204/E829/F1DC/F1F7/F248]
Fehlermeldung bereitstellen
E645  20 C7 E6  JSR $E6C7  Fehlermeldung im Puffer erzeugen
----------------------------------------------------------------------
[A4AA/D021/E641/F01F]
Fehlermeldung aktivieren
E648  20 BD C1  JSR $C1BD  Eingabepuffer für Befehlsstring löschen
E64B  A9 00     LDA #$00   Zurückschreiben der BAM durch setzen des
E64D  8D F9 02  STA $02F9  Flags verhindern
E650  20 2C C1  JSR $C12C  LED blinken lassen
E653  20 DA D4  JSR $D4DA  Kanäle schließen
E656  A9 00     LDA #$00   Zeiger auf Position im Befehlsstring
E658  85 A3     STA $A3    im Eingabepuffer zurücksetzen
E65A  A2 45     LDX #$45   Stackpointer wieder neu
E65C  9A        TXS        setzen
E65D  A5 84     LDA $84    Standard-Sekundäradresse
E65F  29 0F     AND #$0F   herstellen und
E661  85 83     STA $83    merken
E663  C9 0F     CMP #$0F   Vergleich mit Kanal 15
E665  F0 31     BEQ $E698  ist es der Kommandokanal ?
E667  78        SEI        nein, Diskcontroller abstellen
E668  A5 79     LDA $79    Flag für 'Listen aufgetreten'
E66A  D0 1C     BNE $E688  aktiv ?
E66C  A5 7A     LDA $7A    nein, Flag für 'Talk aufgetreten'
E66E  D0 10     BNE $E680  aktiv ?
E670  A6 83     LDX $83    nein, Sekundäradresse holen
E672  BD 2B 02  LDA $022B,X und dazugehörigen Kanalzustand
E675  C9 FF     CMP #$FF   testen
```

```
E677    F0 1F       BEQ $E698       ist der Kanal aktiv ?
E679    29 0F       AND #$0F        ja, Kanalnummer herstellen
E67B    85 82       STA $82         und speichern
E67D    4C 8E E6    JMP $E68E       zur Warteschleife
E680[1] 20 EB D0    JSR $D0EB       Kanalnummer holen
E683    EA          NOP             Bereich mit Leerschritten
E684    EA          NOP             [entsteht durch Modifizierung]
E685    EA          NOP             [des 1541 ROM]
E686    D0 06       BNE $E68E       immer Sprung nach $E68E
E688[1] 20 07 D1    JSR $D107       Schreibkanal holen
E68B    EA          NOP             Bereich mit Leerschritten
E68C    EA          NOP             [entsteht durch Modifizierung]
E68D    EA          NOP             [des 1541 ROM]
E68E[2] 20 25 D1    JSR $D125       aktuellen Dateityp feststellen
E691    C9 04       CMP #$04        auf relative Datei prüfen
E693    B0 03       BCS $E698       liegt relative Datei vor ?
E695    20 27 D2    JSR $D227       nein, alle Kanäle freigeben
E698[5] 4C 6B 83    JMP $836B       zur Befehlswarteschleife
----------------------------------------------------------------------
[E6EA/E6F4]
Wandelt einen Binärwert in A in BCD-Wert um
E69B    AA          TAX             Binärwert merken
E69C    A9 00       LDA #$00        Akku zurücksetzen
E69E    F8          SED             [Fehler siehe Kapitel 7.1.5]
E69F[1] E0 00       CPX #$00        Binärwert mit Null vergleichen
E6A1    F0 07       BEQ $E6AA       identisch ?
E6A3    18          CLC             Addition vorbereiten
E6A4    69 01       ADC #$01        X-mal eins im
E6A6    CA          DEX             BCD-Modus addieren
E6A7    4C 9F E6    JMP $E69F       weiterzählen, bis X gleich Null wird
E6AA[1] D8          CLD             Dezimalmodus ausschalten
----------------------------------------------------------------------
[E6D1]
BCD-Zahl in zwei ASCII-Zeichen umwandeln
E6AB    AA          TAX             BCD-Wert merken
E6AC    4A          LSR A           höherwertiges
E6AD    4A          LSR A           Nibble isoliern, damit
E6AE    4A          LSR A           erste Ziffer der
E6AF    4A          LSR A           BCD-Zahl bereitstellen
E6B0    20 B4 E6    JSR $E6B4       und in ASCII-Wert umrechnen
E6B3    8A          TXA             Orginalwert wieder holen
```

```
E6B4  29 0F     AND #$0F       und zweite BCD-Ziffer isolieren
E6B6  09 30     ORA #$30       in ASCII-Zeichen umwandeln
E6B8  91 A5     STA ($A5),Y    und in den aktuellen Puffer schreiben
E6BA  C8        INY            Zeiger auf nächstes Byte im Puffer
E6BB  60        RTS            zurück zur aufrufenden Routine
----------------------------------------------------------------------
[C150/E0D6]
Fehlermeldung '00 OK' bereitstellen
E6BC  20 23 C1  JSR $C123      Fehlerflags zurücksetzen
E6BF  A9 00     LDA #$00       Fehlernummer für 'ok'
----------------------------------------------------------------------
[D24A/EBD7]
Fehlermeldung mit Spur,Sektor =0 ausgeben
E6C1  A0 00     LDY #$00       Spurnummer und
E6C3  84 80     STY $80        Sektornummer
E6C5  84 81     STY $81        löschen
----------------------------------------------------------------------
[C1A7/E63B/E645/EFCB]
Fehlermeldung im Puffer erzeugen (Nummer im Akku)
E6C7  A0 00     LDY #$00       Zeiger auf Position im Puffer setzen
E6C9  A2 D5     LDX #$D5       Pufferadresse des Fehlermeldungs-
E6CB  86 A5     STX $A5        puffers ($02D5) im
E6CD  A2 02     LDX #$02       Zeiger $A5/$A6
E6CF  86 A6     STX $A6        merken
E6D1  20 AB E6  JSR $E6AB      Fehlernummer in Puffer schreiben
E6D4  A9 2C     LDA #$2C       ',' Komma
E6D6  91 A5     STA ($A5),Y    nach Fehlernummer in Puffer übertragen
E6D8  C8        INY            Pufferzeiger auf nächstes Byte setzen
E6D9  AD D5 02  LDA $02D5      erste Ziffer der Fehlernummer
E6DC  8D 43 02  STA $0243      ins Ausgaberegister kopieren
E6DF  8A        TXA            Fehlernummer wieder holen
E6E0  20 06 E7  JSR $E706      Fehlermeldung im Klartext in Puffer
E6E3  A9 2C     LDA #$2C       schreiben und nachfolgendes
E6E5  91 A5     STA ($A5),Y    Komma setzen
E6E7  C8        INY            Pufferzeiger auf nächstes Byte setzen
E6E8  A5 80     LDA $80        Nummer der Spur, wo Fehler auftrat
E6EA  20 9B E6  JSR $E69B      in ASCII umwandeln und in Puffer
E6ED  A9 2C     LDA #$2C       ',' Komma
E6EF  91 A5     STA ($A5),Y    in Puffer als Trennzeichen setzen
E6F1  C8        INY            Pufferzeiger auf nächstes Byte
E6F2  A5 81     LDA $81        Nummer des Sektors, wo Fehler auftrat
```

```
E6F4  20 9B E6   JSR $E69B       in ASCII umwandeln und in Puffer
E6F7  88         DEY             Länge der
E6F8  98         TYA             Fehlermeldung
E6F9  18         CLC             im Puffer
E6FA  69 D5      ADC #$D5        errechnen und
E6FC  8D 49 02   STA $0249       merken
E6FF  E6 A5      INC $A5         Pufferzeiger $A5/$A6 auf zweites Zeichen
E701  A9 88      LDA #$88        Flag für 'Fertig zum Ausgeben'
E703  85 F7      STA $F7         setzen und
E705  60         RTS             zurück zur aufrufenden Routine
------------------------------------------------------------------------
[E6E0/E75F]
Fehlermeldung im Klartext in Fehlerpuffer schreiben
E706  AA         TAX             Fehlernummer merken
E707  A5 86      LDA $86         die Werte des Zwischenspeichers,
E709  48         PHA             der benutzt wird,
E70A  A5 87      LDA $87         werden gerettet, da diese Adressen
E70C  48         PHA             von der Routine benötigt werden
E70D  A9 FC      LDA #$FC        Adresse für Beginn
E70F  85 86      STA $86         der Texttabelle ($E4FC)
E711  A9 E4      LDA #$E4        in Zeiger $86/$87
E713  85 87      STA $87         setzen
E715  8A         TXA             Fehlernummer wieder holen
E716  A2 00      LDX #$00        Pufferzeiger initialisieren
E718  C1 86      CMP ($86,X)     Nummer mit Texttabelle vergleichen
E71A  F0 21      BEQ $E73D       identisch ?
E71C  48         PHA             nein, Fehlernummer merken
E71D  20 75 E7   JSR $E775       Pufferzeiger erhöhen
E720  90 05      BCC $E727       immer Sprung nach $E727
E722  20 75 E7   JSR $E775       Pufferzeiger erhöhen
E725  90 FB      BCC $E722       immer Sprung nach $E727
E727  A5 87      LDA $87         High-Byte des Textzeigers holen
E729  C9 E6      CMP #$E6        und auf Endwert prüfen
E72B  90 08      BCC $E735       ist Ende der Tabelle erreicht ?
E72D  D0 0A      BNE $E739       ja, ist gleiche Speicherseite erreicht ?
E72F  A9 0A      LDA #$0A        ja, Low-Byte des Textzeigers
E731  C5 86      CMP $86         mit Endwert vergleichen
E733  90 04      BCC $E739       ist Ende der Fehlertabelle erreicht ?
E735  68         PLA             ja, Fehlernummer wieder holen
E736  4C 18 E7   JMP $E718       Fehlernummer suchen
E739  68         PLA             Fehlernummer wieder holen
```

```
E73A    4C 4D E7   JMP $E74D       Ende
E73D[2] 20 67 E7   JSR $E767       Byte aus Fehlertext holen
E740    90 FB      BCC $E73D       ist Anfang-Flag gesetzt ?
E742[1] 20 54 E7   JSR $E754       ja, Zeichen in Puffer schreiben
E745    20 67 E7   JSR $E767       Byte aus Fehlertext holen
E748    90 F8      BCC $E742       ist Ende-Flag gesetzt ?
E74A    20 54 E7   JSR $E754       ja, Zeichen in Puffer schreiben
E74D[1] 68         PLA             Zeropagewerte wieder
E74E    85 87      STA $87         holen und ursprüngliche
E750    68         PLA             Werte wieder
E751    85 86      STA $86         herstellen
E753    60         RTS             zurück zur aufrufenden Routine
-----------------------------------------------------------------------
```

[E742/E74A]
ASCII-Zeichen in Puffer schreiben
Nicht-ASCII-Zeichen werden als Fehlernummer interpretiert

```
E754    C9 20      CMP #$20        mit Leerzeichen vergleichen
E756    B0 0B      BCS $E763       ist Zeichen größer als Leerzeichen ?
E758    AA         TAX             nein, Zeichen als Fehlernummer merken
E759    A9 20      LDA #$20        und Leerzeichen in
E75B    91 A5      STA ($A5),Y     aktuelle Pufferposition schreiben
E75D    C8         INY             Pufferzeiger auf nächstes Zeichen setzen
E75E    8A         TXA             und Fehlernummer wieder holen
E75F    20 06 E7   JSR $E706       Klartext der Fehlermeldung in Puffer
E762    60         RTS             zurück zur aufrufenden Routine
E763[1] 91 A5      STA ($A5),Y     ASCII-Zeichen in Puffer schreiben
E765    C8         INY             und Pufferzeiger auf nächste Position
E766    60         RTS             zurück zur aufrufenden Routine
-----------------------------------------------------------------------
```

[E73D/E745]
Ein Zeichen des Fehlertexts aus der Texttabelle holen

```
E767    E6 86      INC $86         Textzeiger auf nächstes Zeichen
E769    D0 02      BNE $E76D       ist ein Übertrag entstanden ?
E76B    E6 87      INC $87         ja, High-Byte korregieren
E76D[2] A1 86      LDA ($86,X)     Zeichen aus Textttabelle holen
E76F    0A         ASL A           Bit7 ins Carry
E770    A1 86      LDA ($86,X)     Orginal-Zeichen nochmal holen
E772    29 7F      AND #$7F        Bit7 ausblenden
E774    60         RTS             zurück zur aufrufenden Routine
-----------------------------------------------------------------------
```

```
[E71D/E722]
aktuelles Byte aus Tabelle holen
E775  20 6D E7   JSR $E76D      Zeichen aus Tabelle holen
E778  E6 86      INC $86        Textzeiger auf nächstes Byte richten
E77A  D0 02      BNE $E77E      ist ein Übertrag entstanden ?
E77C  E6 87      INC $87        ja, High-Byte korregieren
E77E  60         RTS            zurück zur aufrufenden Routine
-----------------------------------------------------------------
[Teil wird im 1571 DOS nicht verwendet]
E77F  60         RTS            Frühere 1541-ROM-Versionen enthielten
E780  60         RTS            hier eine Autoboot-Routine
E781  EA ...     NOP            Der eventuelle Einsprung in
E7A1  ... EA     NOP            diese Routine wird bei der 1571 hiermit
E7A2  60         RTS            abgefangen und beendet
-----------------------------------------------------------------
[Einsprung durch Routine C146/E7A8:A611,A619]
Routine für den &-Befehl (Autostart-Programme)
E7A3  20 FE A5   JSR $A5FE      Patch (=Korrektur) für 1571 DOS
E7A6  EA         NOP            [Leerschritte, da 1541 ROM]
E7A7  EA         NOP            [modifiziert wurde]
E7A8  20 58 F2   JSR $F258      keine Funktion (rts)
E7AB  AD 78 02   LDA $0278      Zahl der Namen für die Dateinennung
E7AE  48         PHA            merken
E7AF  A9 01      LDA #$01       Arbeit auf erste
E7B1  8D 78 02   STA $0278      Datei begrenzen
E7B4  A9 FF      LDA #$FF       Flag für Eintrag
E7B6  85 86      STA $86        löschen
E7B8  20 4F C4   JSR $C44F      Dateieintrag im Directory suchen
E7BB  AD 80 02   LDA $0280      Flag für Suchergebnis (Spurnummer)
E7BE  D0 05      BNE $E7C5      Dateieintrag gefunden ?
E7C0  A9 39      LDA #$39       Fehlermeldung
E7C2  20 C8 C1   JSR $C1C8      '39 File Not Found' ausgeben
E7C5  68         PLA            Zahl der Dateinamen wieder holen
E7C6  8D 78 02   STA $0278      und zurücksetzen
E7C9  AD 80 02   LDA $0280      Spur des ersten Sektors der Datei
E7CC  85 80      STA $80        übertragen
E7CE  AD 85 02   LDA $0285      Nummer des ersten Sektors
E7D1  85 81      STA $81        übertragen
E7D3  A9 03      LDA #$03       Kennzeichen für USR-Datei
E7D5  20 77 D4   JSR $D477      Datei eröffnen; ersten Sektor einlesen
E7D8  A9 00      LDA #$00       Prüfsumme
```

```
E7DA    85 87      STA $87        löschen
E7DC    20 39 E8   JSR $E839      Speicherstartadresse aus
E7DF    85 88      STA $88        dem Puffer holen
E7E1    20 4B E8   JSR $E84B      und im Zeiger $88/$89
E7E4    20 39 E8   JSR $E839      merken; dabei jeweils
E7E7    85 89      STA $89        den geholten Wert
E7E9    20 4B E8   JSR $E84B      zur Prüfsumme hinzurechnen
E7EC    A5 86      LDA $86        Flag für 'Startadresse gelesen'
E7EE    F0 0A      BEQ $E7FA      gesetzt ?
E7F0    A5 88      LDA $88        nein, Startadresse Low-Byte
E7F2    48         PHA            merken und
E7F3    A5 89      LDA $89        Startadresse High-Byte
E7F5    48         PHA            merken
E7F6    A9 00      LDA #$00       Flag für 'Startadresse gelesen'
E7F8    85 86      STA $86        setzen
E7FA[1] 20 39 E8   JSR $E839      Zahl der folgenden Datenbytes
E7FD    85 8A      STA $8A        aus dem Puffer holen und merken
E7FF    20 4B E8   JSR $E84B      Prüfsumme aktualisieren
E802[1] 20 39 E8   JSR $E839      Datenbyte aus Puffer holen
E805    A0 00      LDY #$00       Zeiger in Speicher initialisieren
E807    91 88      STA ($88),Y    und Byte abspeichern
E809    20 4B E8   JSR $E84B      Byte in Prüfsumme übernehmen
E80C    A5 88      LDA $88        Speicheradresse (Low-Byte)
E80E    18         CLC            die Adresse, wo Daten
E80F    69 01      ADC #$01       abgelegt werden sollen
E811    85 88      STA $88        um eins erhöhen
E813    90 02      BCC $E817      ist ein Übertrag entstanden ?
E815    E6 89      INC $89        ja, High-Byte korregieren
E817[1] C6 8A      DEC $8A        Zähler für Gesamtzahl der Datenbytes
E819    D0 E7      BNE $E802      alle Bytes im Speicher ?
E81B    20 35 CA   JSR $CA35      ja, Prüfsumme aus Puffer holen
E81E    A5 85      LDA $85        Prüfsumme mit
E820    C5 87      CMP $87        errechnetem Wert vergleichen
E822    F0 08      BEQ $E82C      identisch ?
E824    20 3E DE   JSR $DE3E      Spur- und Sektor holen
E827    A9 50      LDA #$50       Fehlermeldung
E829    20 45 E6   JSR $E645      '50 Record Not Present' ausgeben
E82C[1] A5 F8      LDA $F8        Flag für EOI (letztes Zeichen) holen
E82E    D0 A8      BNE $E7D8      wurde letztes Zeichen bearbeitet ?
E830    68         PLA            ja, Startadresse des Programms
E831    85 89      STA $89        wieder holen
```

```
E833  68          PLA              und in Zeiger $88/$89
E834  85 88       STA $88          einrichten
E836  6C 88 00    JMP ($0088)      über Zeiger Sprung zu Programm
--------------------------------------------------------------------------
[E7DC/E7E4/E7FA/E802]
Byte aus Puffer holen
E839  20 35 CA    JSR $CA35        Byte aus Sektor holen
E83C  A5 F8       LDA $F8          Flag für EOI testen
E83E  D0 08       BNE $E848        war das das letzte Zeichen ?
E840  20 3E DE    JSR $DE3E        ja, Spur und Sektor setzen
E843  A9 51       LDA #$51         Fehlermeldung
E845  20 45 E6    JSR $E645        '51 Overflow in Record' ausgeben
E848  A5 85       LDA $85          letztes Zeichen holen
E84A  60          RTS              zurück zur aufrufenden Routine
--------------------------------------------------------------------------
[E7E1/E7E9/E7FF/E809]
Prüfsumme aktualisieren
E84B  18          CLC              neues Byte
E84C  65 87       ADC $87          zu bisherigen Werten addieren
E84E  69 00       ADC #$00         den überlauf auch dazurechnen
E850  85 87       STA $87          und neuen Prüfsummenwert merken
E852  60          RTS              zurück zur aufrufenden Routine
--------------------------------------------------------------------------
[9DC1]
Flag für ATN vom seriellen Bus empfangen wird gesetzt
E853  AD 01 18    LDA $1801        [Fehler siehe Kapitel 7.1.5]
E856  A9 01       LDA #$01         Flag für 'ATN empfangen'
E858  85 7C       STA $7C          setzen
E85A  60          RTS              zurück zur aufrufenden Routine
--------------------------------------------------------------------------
[A7BD/EA56/EA68/E8D7:A9B6]
Routine zum Steuern des seriellen Bus
E85B  78          SEI              Diskcontroller abschalten
E85C  A9 00       LDA #$00         Flags mit Null löschen :
E85E  85 7C       STA $7C          Flag für 'ATN empfangen'
E860  85 79       STA $79          Flag für Listen
E862  85 7A       STA $7A          Flag für Talk
E864  A2 45       LDX #$45         Stackpointer neu
E866  9A          TXS              setzen
E867  A9 80       LDA #$80         Flags mit $80 (Bit7 aktiv) löschen :
E869  85 F8       STA $F8          Flag für EOI (Ende der Übertragung)
```

```
E86B  85 7D     STA $7D     Flag für ATN-Modus
E86D  20 B7 E9  JSR $E9B7   Clock auf High setzen
E870  20 A5 E9  JSR $E9A5   Data Leitung auf Low legen
E873  AD 00 18  LDA $1800   Bus-Steuerregister holen
E876  09 10     ORA #$10    ATN Request löschen
E878  8D 00 18  STA $1800   und auf Bus geben
E87B  AD 00 18  LDA $1800   Busstatus wieder holen
E87E  10 57     BPL $E8D7   ist ATN gesetzt ?
E880  29 04     AND #$04    nein, Clock Leitung maskieren
E882  D0 F7     BNE $E87B   ist Clock gesetzt ?
E884  20 C9 E9  JSR $E9C9   ja, Kommandowort vom Bus einlesen
E887  C9 3F     CMP #$3F    mit Unlisten vergleichen
E889  D0 06     BNE $E891   identisch ?
E88B  A9 00     LDA #$00    ja, Flag für Listen
E88D  85 79     STA $79     löschen
E88F  F0 71     BEQ $E902   immer Sprung nach $E902
E891  C9 5F     CMP #$5F    mit Untalk vergleichen
E893  D0 06     BNE $E89B   identisch ?
E895  A9 00     LDA #$00    ja, Flag für Talk
E897  85 7A     STA $7A     löschen
E899  F0 67     BEQ $E902   immer Sprung nach $E902
E89B  C5 78     CMP $78     Kennzeichen für Talkadresse
E89D  D0 0A     BNE $E8A9   Soll Talkadresse empfangen werden ?
E89F  A9 01     LDA #$01    ja, Flag für Talk
E8A1  85 7A     STA $7A     setzen
E8A3  A9 00     LDA #$00    Flag für Listen
E8A5  85 79     STA $79     löschen
E8A7  F0 29     BEQ $E8D2   immer Sprung nach $E8D2
E8A9  C5 77     CMP $77     Kennzeichen für Listenadresse
E8AB  D0 0A     BNE $E8B7   Soll Listenadresse empfangen werden ?
E8AD  A9 01     LDA #$01    ja, Flag für Listen
E8AF  85 79     STA $79     setzen
E8B1  A9 00     LDA #$00    Flag für Talk
E8B3  85 7A     STA $7A     löschen
E8B5  F0 1B     BEQ $E8D2   immer Sprung nach $E8D2
E8B7  AA        TAX         Kommando merken
E8B8  29 60     AND #$60    Kommadobits isolieren
E8BA  C9 60     CMP #$60    auf Open testen
E8BC  D0 3F     BNE $E8FD   identisch ?
E8BE  8A        TXA         ja, Kommandowort wieder holen
E8BF  85 84     STA $84     und merken
```

```
E8C1     29 0F      AND #$0F     reine Kanalnummer herstellen
E8C3     85 83      STA $83      und merken
E8C5     A5 84      LDA $84      Kommandowort wieder holen
E8C7     29 F0      AND #$F0     Adressbits ausblenden
E8C9     C9 E0      CMP #$E0     mit Close-Kommando vergleichen
E8CB     D0 35      BNE $E902    identisch ?
E8CD     58         CLI          ja, Diskcontroller einschalten
E8CE     20 C0 DA   JSR $DAC0    Close aufrufen
E8D1     78         SEI          Diskcontroller wieder ausschalten
E8D2[2]  2C 00 18   BIT $1800    ATN-Bit prüfen
E8D5     30 AD      BMI $E884    ATN aktiv ?; wenn ja warten
E8D7[3]  A9 00      LDA #$00     nein,
E8D9     85 7D      STA $7D      Flag für Kommandomodus löschen
E8DB     AD 00 18   LDA $1800    Bus-Steuerregister
E8DE     29 EF      AND #$EF     ATN löschen
E8E0     8D 00 18   STA $1800    und auf Bus ausgeben
E8E3     A5 79      LDA $79      Flag für Listen
E8E5     F0 06      BEQ $E8ED    Flag gesetzt ?
E8E7     20 2E EA   JSR $EA2E    Daten vom Bus in Puffer übernehmen
E8EA     4C 6B 83   JMP $836B    auf nächstes Kommandowort warten
E8ED[1]  A5 7A      LDA $7A      Flag für Talk
E8EF     F0 09      BEQ $E8FA    aktiv ?
E8F1     20 9C E9   JSR $E99C    Data auf High setzen
E8F4     20 AE E9   JSR $E9AE    Clock auf Low setzen
E8F7     20 09 E9   JSR $E909    Daten aus Puffer an Bus übergeben
E8FA[1]  4C 4E EA   JMP $EA4E    auf nächstes Kommandowort warten
E8FD[1]  A9 10      LDA #$10     Kein Talk- oder Listenbefehl
E8FF     8D 00 18   STA $1800    Datenleitungen zurücksetzen
E902[4]  2C 00 18   BIT $1800    ATN prüfen
E905     10 D0      BPL $E8D7    ist ATN zurückgesetzt ?
E907     30 F9      BMI $E902    nein, weiter warten bis Kommandoende
-------------------------------------------------------------------------
[E8F7]
Daten nach Talk-Aufruf senden
E909     78         SEI          Diskcontroller ausschalten
E90A     20 EB D0   JSR $D0EB    freien Kanal suchen und öffnen
E90D     B0 06      BCS $E915    ist ein Kanal frei ?
E90F[1]  A6 82      LDX $82      ja, aktuelle Kanalnummer holen
E911     B5 F2      LDA $F2,X    und dazugehörigen Status holen
E913     30 01      BMI $E916    ist Kanal auf Lesen ?
E915[1]  60         RTS          nein, züruck zur aufrufenden Routine
```

```
E916[1] 20 59 EA   JSR $EA59      ATN-Leitung prüfen
E919    20 C0 E9   JSR $E9C0      Wert vom Busregister lesen
E91C    29 01      AND #$01       und Data-Eingang holen
E91E    08         PHP            Zustand der Data-Leitung merken
E91F    20 B7 E9   JSR $E9B7      Clock-Ausgang auf Low setzen
E922    28         PLP            Zustand der Data-Leitung wieder holen
E923    F0 12      BEQ $E937      war Data gesetzt ?
E925[1] 20 59 EA   JSR $EA59      ja, auf ATN-Kommandomodus testen
E928    20 C0 E9   JSR $E9C0      Wert von Busregister holen
E92B    29 01      AND #$01       Data-Leitung isolieren
E92D    D0 F6      BNE $E925      warten bis Data auf Low ist
E92F    A6 82      LDX $82        Nummer der internen Kanals
E931    B5 F2      LDA $F2,X      zugehörigen Status holen
E933    29 08      AND #$08       und Flag für EOI testen
E935    D0 14      BNE $E94B      wurde letztes Zeichen gesendet ?
E937[2] 20 59 EA   JSR $EA59      ja, auf ATN-Modus testen
E93A    20 C0 E9   JSR $E9C0      Wert von Busregister holen
E93D    29 01      AND #$01       und Data Leitung prüfen
E93F    D0 F6      BNE $E937      warten bis Data auf Low ist
E941[1] 20 59 EA   JSR $EA59      auf ATN-Kommandomodus testen
E944    20 C0 E9   JSR $E9C0      Wert von Busregister holen
E947    29 01      AND #$01       und Data Leitung isolieren
E949    F0 F6      BEQ $E941      warten bis Data Eingang auf High
E94B[2] 20 AE E9   JSR $E9AE      Clock-Ausgang auf High setzen
E94E    20 59 EA   JSR $EA59      auf ATN-Modus prüfen
E951    20 C0 E9   JSR $E9C0      Wert von Busregister holen
E954    29 01      AND #$01       und Data analysieren
E956    D0 F3      BNE $E94B      warten bis Data auf Low ist
E958    A9 08      LDA #$08       Zahl der Bits pro Byte in
E95A    85 98      STA $98        Zähler setzen
E95C[1] 20 C0 E9   JSR $E9C0      Wert von Busregister holen
E95F    29 01      AND #$01       und Data Leitung prüfen
E961    D0 36      BNE $E999      ist Data auf Low ?
E963    A6 82      LDX $82        ja, aktuelle Kanalnummer
E965    BD 3E 02   LDA $023E,X    dazugehöriges Datenbyte holen
E968    6A         ROR A          und erstes Bit davon ins Carry
E969    9D 3E 02   STA $023E,X    Rest wieder merken
E96C    B0 05      BCS $E973      ist Bit auf 1 ?
E96E    20 A5 E9   JSR $E9A5      nein, Data-Leitung auf High setzen
E971    D0 03      BNE $E976      immer Sprung nach $E976
E973[1] 20 9C E9   JSR $E99C      Data Leitung auf Low setzen
```

```
E976[1] 20 B7 E9   JSR $E9B7      Clock Leitung auf Low setzen
E979  A5 23        LDA $23        Flag für Busmodus testen
E97B  D0 03        BNE $E980      ist Bus im 1540 Modus ?
E97D  20 F3 FE     JSR $FEF3      nein, 42 Taktzyklen Verzögerung
E980[1] 20 FB FE   JSR $FEFB      Data auf Low, Clock auf High setzen
E983  C6 98        DEC $98        Zahl der zu übertragenden Bits
E985  D0 D5        BNE $E95C      ist Byte schon gesendet ?
E987[1] 20 59 EA   JSR $EA59      ja, auf ATN-Modus testen
E98A  20 C0 E9     JSR $E9C0      Wert von Busregister holen
E98D  29 01        AND #$01       und Data Leitung prüfen
E98F  F0 F6        BEQ $E987      ist Data gesetzt ?
E991  58           CLI            ja, Diskcontroller einschalten
E992  20 AA D3     JSR $D3AA      nächstes Datenbyte aus Puffer holen
E995  78           SEI            Diskcontroller wieder ausschalten
E996  4C 0F E9     JMP $E90F      und auf Bus ausgeben
E999[1] 4C 4E EA   JMP $EA4E      auf nächstes Kommando warten
-----------------------------------------------------------------
[817B/8291/82D8/8300/E8F1/E973/E9D7/E9FA/FEFE]
Data Leitung auf logisch Low setzen (physikalisch High)
E99C  AD 00 18     LDA $1800      Bus-Steuerregister lesen
E99F  29 FD        AND #$FD       Bit für Data-Leitung
E9A1  8D 00 18     STA $1800      löschen
E9A4  60           RTS            zurück zur aufrufenden Routine
-----------------------------------------------------------------
[80E6/828C/82F8/833C/E870/E96E/E9F2/EA28]
Data Leitung auf logisch High setzen (physikalisch Low)
E9A5  AD 00 18     LDA $1800      Bus-Steuerregister holen
E9A8  09 02        ORA #$02       und Bit für Data-Leitung
E9AA  8D 00 18     STA $1800      setzen
E9AD  60           RTS            zurück zur aufrufenden Routine
-----------------------------------------------------------------
[817E/822C/E8F4/E94B/FEFB]
Clock Leitung auf logisch High setzen (physikalisch Low)
E9AE  AD 00 18     LDA $1800      Bus-Steuerregister holen
E9B1  09 08        ORA #$08       und Bit für Clock-Leitung
E9B3  8D 00 18     STA $1800      setzen
E9B6  60           RTS            zurück zur aufrufenden Routine
-----------------------------------------------------------------
```

```
[80E3/8200/829E/8438/E86D/E91F/E976]
Clock Leitung auf logisch Low setzen (physikalisch High)
E9B7  AD 00 18   LDA $1800      Bus-Steuerregister holen
E9BA  29 F7      AND #$F7       und Bit für Clock-Leitung
E9BC  8D 00 18   STA $1800      löschen
E9BF  60         RTS            zurück zur aufrufenden Routine
------------------------------------------------------------------
[819C/81FA/8209/821B/8225/8232/827A/82B5/82D1/82EF/8306/8331/E919/E928]
[E93A/E944/E951/E95C/E98A/E9C6/E9D0/E9E9/EA00/EA1D]
Wert vom Bus lesen (Wert entprellen)
E9C0  AD 00 18   LDA $1800      Steuerregister holen
E9C3  CD 00 18   CMP $1800      Nochmal holen und mit letztem Wert
E9C6  D0 F8      BNE $E9C0      vergleichen; Wert konstant geblieben ?
E9C8  60         RTS            ja, zurück zur aufrufenden Routine
------------------------------------------------------------------
[E884/EA44/E9DF:FF2C]
Daten nach Listen Aufruf empfangen
E9C9  A9 08      LDA #$08       Zahl der Bits pro Datenbyte
E9CB  85 98      STA $98        Zähler initialisieren
E9CD[1] 20 59 EA JSR $EA59      ATN testen
E9D0  20 C0 E9   JSR $E9C0      Bus-Steuerregister holen
E9D3  29 04      AND #$04       und Clock-Leitung testen
E9D5  D0 F6      BNE $E9CD      Clock aktiv ?
E9D7  20 9C E9   JSR $E99C      ja, Data Leitung aktivieren
E9DA  A9 01      LDA #$01       [Fehler siehe 7.1.5]
E9DC  4C 20 FF   JMP $FF20      warten bis Data auf Low; Timer setzen
E9DF[1] 20 59 EA JSR $EA59      ATN testen
E9E2  AD 0D 18   LDA $180D      Interruptflags holen
E9E5  29 40      AND #$40       und Flag für Timer 1 testen
E9E7  D0 09      BNE $E9F2      ist Timer abgelaufen ?
E9E9  20 C0 E9   JSR $E9C0      nein, Wert von Busregister holen
E9EC  29 04      AND #$04       Clock Eingang isolieren
E9EE  F0 EF      BEQ $E9DF      ist Clock gesetzt ?
E9F0  D0 19      BNE $EA0B      ja, immer Sprung nach $EA0B
E9F2[1] 20 A5 E9 JSR $E9A5      Data-Leitung auf High setzen
E9F5  A2 0A      LDX #$0A       Verzögerungszähler setzen
E9F7[1] CA       DEX            Verzögerung von 51 Taktzyklen erzeugen
E9F8  D0 FD      BNE $E9F7      ist Verzögerung abgelaufen ?
E9FA  20 9C E9   JSR $E99C      ja, Data Leitung auf Low setzen
E9FD[1] 20 59 EA JSR $EA59      Bus auf ATN-Kommando testen
EA00  20 C0 E9   JSR $E9C0      Bus-Steuerregister holen
```

```
EA03    29 04      AND #$04     Clock-Leitung prüfen
EA05    F0 F6      BEQ $E9FD    Clock gesetzt ?
EA07    A9 00      LDA #$00     ja, Daten zu Ende
EA09    85 F8      STA $F8      EOI Flag setzen
EA0B[3] AD 00 18   LDA $1800    Steuerregister wieder holen
EA0E    49 01      EOR #$01     Datenbit auf Orginalwert korregieren
EA10    4A         LSR A        und im Carry merken
EA11    29 02      AND #$02     Clock Leitung testen (versetzt mit LSR)
EA13    D0 F6      BNE $EA0B    Clock gesetzt, Daten in Ordnung ?
EA15    EA         NOP          ja, Leerbereich
EA16    EA         NOP          (kann nicht für eigene Daten
EA17    EA         NOP          verwendet werden)
EA18    66 85      ROR $85      Datenbit in Zwischenspeicher schieben
EA1A[1] 20 59 EA   JSR $EA59    ATN testen
EA1D    20 C0 E9   JSR $E9C0    Bus-Steuerregister lesen
EA20    29 04      AND #$04     Clock Leitung prüfen
EA22    F0 F6      BEQ $EA1A    ist Clock gesetzt ?
EA24    C6 98      DEC $98      ja, Zähler für Zahl der Datenbits setzen
EA26    D0 E3      BNE $EA0B    bereits acht Bits gelesen ?
EA28    20 A5 E9   JSR $E9A5    ja, Data Leitung auf low setzen
EA2B    A5 85      LDA $85      Datenbyte übernehmen
EA2D    60         RTS          zurück zur aufrufenden Routine
------------------------------------------------------------------------
[EA4B/E8E7]
Daten vom Bus übernehmen und im aktuellen Puffer ablegen
EA2E    78         SEI          Diskcontroller ausschalten
EA2F    20 07 D1   JSR $D107    Schreibkanal belegen
EA32    B0 05      BCS $EA39    freier Kanal gefunden ?
EA34    B5 F2      LDA $F2,X    ja, zugehöriger Kanalstatus lesen
EA36    6A         ROR A        ist der Kanal
EA37    B0 0B      BCS $EA44    aktiv ?
EA39[1] A5 84      LDA $84      ja, Sekundäradresse holen
EA3B    29 F0      AND #$F0     und Kommandobits isolieren
EA3D    C9 F0      CMP #$F0     mit Open-Kommando vergleichen
EA3F    F0 03      BEQ $EA44    identisch ?
EA41    4C 4E EA   JMP $EA4E    nein, auf nächstes Kommando warten
EA44[2] 20 C9 E9   JSR $E9C9    Byte vom Bus holen
EA47    58         CLI          Diskcontroller wieder einschalten
EA48    20 B7 CF   JSR $CFB7    Datenbyte in Puffer übertragen
EA4B    4C 2E EA   JMP $EA2E    nächstes Byte holen
EA4E[3] A9 00      LDA #$00     Bus-Steuerregister
```

```
EA50  8D 00 18   STA $1800       zurücksetzen
EA53  4C 6B 83   JMP $836B       auf nächstes Kommando warten
---------------------------------------------------------------------
EA56  4C 5B E8   JMP $E85B       unbenutzter Programmrest aus 1541 DOS
---------------------------------------------------------------------
[8199/81F7/8206/8218/8222/822F/82B2/82CE/82E5/8303/832E/8425/85F6]
[869D/8E14/E916/E925/E937/E941/E94E/E987/E9CD/E9DF/E9FD/EA1A]
Testet, ob Kommando übertragen wird
EA59  A5 7D      LDA $7D         Flag für 'ATN aktiv'
EA5B  F0 06      BEQ $EA63       ATN gesetzt ?
EA5D  AD 00 18   LDA $1800       ja, aktueller Zustand der ATN-Leitung
EA60  10 09      BPL $EA6B       ist ATN auch gesetzt ?
EA62  60         RTS             ja, zurück zur aufrufenden Routine
EA63  AD 00 18   LDA $1800       aktueller Zustand der ATN-Leitung holen
EA66  10 FA      BPL $EA62       ist immer noch ATN gesetzt ?
EA68  4C B3 A7   JMP $A7B3       ja, Kommando vom Bus annehmen
EA6B  4C AC A9   JMP $A9AC       ATN zurücksetzen
---------------------------------------------------------------------
[EAB5/EABE/EAC4]
Anzeige von Hardwarefehlern (Endlosschleife) durch LED-Blinken alle 1 sec
EA6E  A2 00      LDX #$00        Blinkzähler setzen
EA70  2C A6 6F   .byte $2C       nächste 2 Bytes überspringen (Bitbefehl)
---------------------------------------------------------------------
[92B7/EB1F]
RAM oder ROM Fehler (Test und Prüfsumme)
EA71  A6 6F      LDX $6f         Blinkzähler holen
EA73  9A         TXS             und merken
EA74  BA         TSX             Blinkwert wieder holen
EA75  A9 08      LDA #$08        LED-Bit (Bit3)
EA77  0D 00 1C   ORA $1C00       im Diskcontrollerregister setzen
EA7A  4C EA FE   JMP $FEEA       und LED aktivieren; weiter bei $EA7D
EA7D  98         TYA             (0) Zählerroutine
EA7E  18         CLC             intialisieren
EA7F  69 01      ADC #$01        Verzögerungszähler
EA81  D0 FC      BNE $EA7F       für ca. 0.3 / 0.1 sec
EA83  88         DEY             je nach 1 oder 2 MHz Takt
EA84  D0 F8      BNE $EA7E       ist Zeit abgelaufen ?
EA86  AD 00 1C   LDA $1C00       ja, Steuerregister holen
EA89  29 F7      AND #$F7        und Bit für 'LED an'
EA8B  8D 00 1C   STA $1C00       ausblenden
EA8E  98         TYA             (0) Zählerroutine
```

```
EA8F[1]  18        CLC          intialisieren
EA90[1]  69 01     ADC #$01     Verzögerungszähler
EA92  D0 FC        BNE $EA90    für ca. 0.3 / 0.1 sec
EA94  88           DEY          je nach 1 oder 2 MHz Takt
EA95  D0 F8        BNE $EA8F    ist Zeit abgelaufen ?
EA97  CA           DEX          Blinkzähler
EA98  10 DB        BPL $EA75    nochmal Blinken ?
EA9A  E0 FC        CPX #$FC     nein, ca. 1 / 0.5 sec warten
EA9C  D0 F0        BNE $EA8E    Zeit abgelaufen ?
EA9E  F0 D4        BEQ $EA74    ja, Blinken wieder von vorne starten
----------------------------------------------------------------------
[89EC/Einsprung über Systemvektor FFFC]
1571                        Reset                          Einsprung
----------------------------------------------------------------------
RAM und ROM testen
EAA0  78           SEI          Diskcontroller abschalten
EAA1  D8           CLD          Arithmetikmodus auf Binär setzen
EAA2  A2 66        LDX #$66     Wert für DDRA
EAA4  4C 10 FF     JMP $FF10    VIA's initialisieren; Sprung nach $EAA7
EAA7  E8           INX          [Fehler siehe 7.1.5]
EAA8  A0 00        LDY #$00     Offsetzähler initialisieren
EAAA  A2 00        LDX #$00     Zeiger auf Zeropagestelle setzen
EAAC[1]  8A        TXA          Nummer der Speicherstelle
EAAD  95 00        STA $00,X    in Speicherstelle schreiben
EAAF  E8           INX          nächstes Speicherzelle anwählen
EAB0  D0 FA        BNE $EAAC    schon Adresse $100 erreicht ?
EAB2[1]  8A        TXA          ja, Soll-Wert der Speicherstelle
EAB3  D5 00        CMP $00,X    mit Ist-Wert vergleichen
EAB5  D0 B7        BNE $EA6E    beide identisch ?
EAB7[1]  F6 00     INC $00,X    ja, Speicherstelleninhalt erhöhen
EAB9  C8           INY          Offsetwert Speicheradresse-Inhalt setzen
EABA  D0 FB        BNE $EAB7    schon alle Werte getestet ?
EABC  D5 00        CMP $00,X    ja, Wert mit Speicherstelle vergleichen
EABE  D0 AE        BNE $EA6E    identisch ?
EAC0  94 00        STY $00,X    ja, Speicherstelle mit (0) löschen
EAC2  B5 00        LDA $00,X    Inhalt wieder holen
EAC4  D0 A8        BNE $EA6E    war Löschen in Ordnung ?
EAC6  E8           INX          ja, nächste Speicherstelle anwählen
EAC7  D0 E9        BNE $EAB2    schon Adresse $100 erreicht ?
EAC9  E6 6F        INC $6F      ja, Blinkzähler für Fehler erhöhen
EACB  A2 80        LDX #$80     Startadresse des
```

EACD	86 76	STX $76	Betriebssystem ROM
EACF	A9 00	LDA #$00	in Zeiger $75/76 auf $8000 setzen
EAD1	85 75	STA $75	setzen
EAD3	A0 02	LDY #$02	Prüfsummenbytes ignorieren
EAD5	18	CLC	ROM-Zeiger
EAD6 [1]	E6 76	INC $76	auf nächste Speicherseite setzen
EAD8 [1]	71 75	ADC ($75),Y	ROM-Wert zur Prüfsumme dazurechnen
EADA	C8	INY	Zeiger auf nächstes Byte richten
EADB	D0 FB	BNE $EAD8	ganze Speicherseite berücksichtigt ?
EADD	CA	DEX	ja, Zahl der Speicherseiten des ROM
EADE	D0 F6	BNE $EAD6	schon ganzes ROM geprüft ?
EAE0	69 FF	ADC #$FF	ja, Prüfsummenwert verrechnen
EAE2	85 76	STA $76	und Ergebnis merken
EAE4	D0 39	BNE $EB1F	ist Fehler vorhanden ?
EAE6	EA	NOP	nein, [Leerbereich]
EAE7	EA	NOP	[der durch Modifzierung]
EAE8	EA	NOP	[des 1541 ROM]
EAE9	EA	NOP	[entstand]
EAEA	A9 01	LDA #$01	Speicherseite 1
EAEC	85 76	STA $76	anwählen
EAEE	E6 6F	INC $6F	Seitennummer in Blinkzähler setzen
EAF0	A2 07	LDX #$07	Zahl der RAM Seiten
EAF2 [2]	98	TYA	Löschwert (Speichernummer)
EAF3	18	CLC	dazu Nummer der
EAF4	65 76	ADC $76	Speicherseite einrechnen
EAF6	91 75	STA ($75),Y	und in Speicher schreiben
EAF8	C8	INY	Zeiger auf nächstes Byte richten
EAF9	D0 F7	BNE $EAF2	ganze Speicherseite gelöscht ?
EAFB	E6 76	INC $76	ja, Zeiger auf nächste Seite setzen
EAFD	CA	DEX	Zahl der RAM Seiten
EAFE	D0 F2	BNE $EAF2	schon ganzes RAM gelöscht
EB00	A2 07	LDX #$07	ja, Zahl der RAM Seiten
EB02 [1]	C6 76	DEC $76	RAM Zeiger auf vorhergehende Seite
EB04 [1]	88	DEY	Zahl der noch zu testenden Seiten
EB05	98	TYA	Nummer der Position
EB06	18	CLC	holen und
EB07	65 76	ADC $76	Seitennummer einrechnen
EB09	D1 75	CMP ($75),Y	mit Löschwert vergleichen
EB0B	D0 12	BNE $EB1F	ist Speicherstelle richtig ?
EB0D	49 FF	EOR #$FF	ja, Werte alle Bits wechseln
EB0F	91 75	STA ($75),Y	und andere Wertigkeit testen

```
EB11  51 75     EOR ($75),Y    Ergebnis prüfen
EB13  91 75     STA ($75),Y    und Speicherstelle löschen (0)
EB15  D0 08     BNE $EB1F      war Test erfolgreich ?
EB17  98        TYA            ja, Prozessorflags für Y-Wert setzen
EB18  D0 EA     BNE $EB04      Ende der Speicherseite ?
EB1A  CA        DEX            ja, nächste Seite anwählen
EB1B  D0 E5     BNE $EB02      schon ganzes RAM getestet ?
EB1D  F0 03     BEQ $EB22      ja, immer Sprung nach $EB22
EB1F  4C 71 EA  JMP $EA71      Hardwarefehler anzeigen
-----------------------------------------------------------------
[EB1D/EB25:A7C4]
Zeropage initialisieren
EB22  4C C0 A7  JMP $A7C0      Stack auf Bereich $0100-$0145 setzen
EB25  AD 00 1C  LDA $1C00      Laufwerksteuerregister holen
EB28  29 F7     AND #$F7       und LED am Laufwerk
EB2A  8D 00 1C  STA $1C00      ausschalten
EB2D  A9 01     LDA #$03       CA1 (ATN) auf positive und CA2 (WP) auf
EB2F  8D 0C 18  STA $180C      negative Flanke triggern
EB32  A9 82     LDA #$82       Flag für 'Interrupt von CA1 aktiv'
EB34  8D 0D 18  STA $180D      löschen
EB37  8D 0E 18  STA $180E      und akktivieren
EB3A  AD 00 18  LDA $1800      Hardwaremäßige Festlegung
EB3D  29 60     AND #$60       der Geräteadresse (durch DIP-Schalter)
EB3F  0A        ASL A          holen und die zwei
EB40  2A        ROL A          signifikanten Bits 5 und 6
EB41  2A        ROL A          in Position 0 und 1
EB42  2A        ROL A          schieben
EB43  09 48     ORA #$48       daraus Geräteadresse für Talker-Betrieb
EB45  85 78     STA $78        generieren und abspeichern
EB47  49 60     EOR #$60       Geräteadresse für Listener-Betrieb
EB49  85 77     STA $77        herstellen und setzen
EB4B  A2 00     LDX #$00       Zeiger auf Pufferzeiger
EB4D  A0 00     LDY #$00       Zeiger auf Tabelle der Highbytes
EB4F  A9 00     LDA #$00       Wert der Low-Bytes
EB51  95 99     STA $99,X      Low-Byte des Pufferzeigers löschen
EB53  E8        INX            Zeiger auf High-Byte setzen
EB54  B9 E0 FE  LDA $FEE0,Y    Highbyte der Pufferadresse holen
EB57  95 99     STA $99,X      und in Zeiger übernehmen
EB59  E8        INX            Zeiger auf nächsten Pufferzeiger
EB5A  C8        INY            Zeiger auf nächstes Highbyte
EB5B  C0 05     CPY #$05       Zahl der Puffer
```

```
EB5D   D0 F0      BNE $EB4F     alle Pufferadressen angelegt ?
EB5F   A9 00      LDA #$00      ja, Low-Byte des Zeigers auf den
EB61   95 99      STA $99,X     Eingabepuffer
EB63   E8         INX           Zeiger auf High-Byte
EB64   A9 02      LDA #$02      Pufferzeiger auf Adresse $200
EB66   95 99      STA $99,X     richten
EB68   E8         INX           Zeiger auf nächstes Byte
EB69   A9 D5      LDA #$D5      Low-Byte des Fehlerpuffers
EB6B   95 99      STA $99,X     setzen
EB6D   E8         INX           Zeiger auf nächstes Byte richten
EB6E   A9 02      LDA #$02      Fehlermeldungspuffer auf Adresse
EB70   95 99      STA $99,X     $02D5 einrichten
EB72   A9 FF      LDA #$FF      Wert für 'Kanal frei'
EB74   A2 12      LDX #$12      Zahl der Sekundäradressen (19)
EB76 1 9D 2B 02   STA $022B,X   Kanal freigeben
EB79   CA         DEX           nächste Sekundäradresse
EB7A   10 FA      BPL $EB76     schon ganze Tabelle angelegt ?
EB7C   A2 05      LDX #$05      ja, Zahl der internen Kanäle (6)
EB7E 1 95 A7      STA $A7,X     1. Puffer freigeben
EB80   95 AE      STA $AE,X     2. Puffer freigeben
EB82   95 CD      STA $CD,X     3. Puffer freigeben
EB84   CA         DEX           nächsten Kanal setzen
EB85   10 F7      BPL $EB7E     schon alle Kanäle berücksichtigt ?
EB87   A9 05      LDA #$05      ja, Eingabepuffer Kanal 4
EB89   85 AB      STA $AB       zuweisen
EB8B   A9 06      LDA #$06      Fehlerpuffer Kanal 5
EB8D   85 AC      STA $AC       zuweisen
EB8F   A9 FF      LDA #$FF      Wert für 'kein Puffer zugeordnet'
EB91   85 AD      STA $AD       in Kanal 6 (1. Puffer
EB93   85 B4      STA $B4       und 2. Puffer)
EB95   A9 05      LDA #$05      Sekundäradresse 16 auf
EB97   8D 3B 02   STA $023B     Kanal 5 leiten
EB9A   A9 84      LDA #$84      Sekundäradresse 15 auf Kanal 4 leiten
EB9C   8D 3A 02   STA $023A     (Status: nur Schreibkanal)
EB9F   A9 0F      LDA #$0F      Flags für Kanalbelegung einrichten
EBA1   8D 56 02   STA $0256     Kanal 0 bis 3 sind freigegeben
EBA4   A9 01      LDA #$01      Flag für 'Schreibkanal'
EBA6   85 F6      STA $F6       für Kanal 4 setzen
EBA8   A9 88      LDA #$88      Flag für 'Lesekanal/ kein EOI'
EBAA   85 F7      STA $F7       für Kanal 5 setzen
EBAC   A9 E0      LDA #$E0      Flags für Pufferbelegung (Bit gesetzt =
```

```
EBAE  8D 4F 02   STA $024F       Puffer belegt) setzen
EBB1  A9 FF      LDA #$FF        dabei Puffer 0 bis 4
EBB3  8D 50 02   STA $0250       freigeben
EBB6  A9 01      LDA #$01        Flag für Zustand der
EBB8  85 1C      STA $1C         Schreibschutzkerbe
EBBA  85 1D      STA $1D         löschen
EBBC  20 63 CB   JSR $CB63       Zeiger auf Sprungtabelle für U-Befehle
EBBF  20 FA CE   JSR $CEFA       Puffer-Kanal-Tabelle initialisieren
EBC2  20 82 FF   JSR $FF82       Diskcontrollerroutine aktivieren
EBC5  A9 22      LDA #$22        Zeiger auf NMI oder Befehl
EBC7  85 65      STA $65         zum Umschalten zwischen 1541 und 1540
EBC9  A9 EB      LDA #$EB        in $65/$66 auf
EBCB  85 66      STA $66         $EB22 setzen
EBCD  A9 06      LDA #$06        Sektorversatz (6)
EBCF  85 69      STA $69         festlegen
EBD1  A9 05      LDA #$05        Zahl der Leseversuche bei Fehlern
EBD3  85 6A      STA $6A         auf 5 setzen
EBD5  A9 73      LDA #$73        DOS Einschaltmeldung
EBD7  20 C1 E6   JSR $E6C1       '73 CBM DOS V3.0 1571' bereitstellen
EBDA  A9 00      LDA #$00        Busleitungen
EBDC  8D 00 18   STA $1800       zurückksetzen
EBDF  A9 1A      LDA #$1A        Eingang/Ausgang Belegung %00011010
EBE1  8D 02 18   STA $1802       festlegen
EBE4  20 86 A7   JSR $A786       CIA 6526 initialisieren
-----------------------------------------------------------------------
[836E/E8EA/EA53]
Warteschleife zur Befehlserkennung
EBE7  58         CLI             Kontroller-/Busroutinen aktivieren
EBE8  AD 00 18   LDA $1800       Bus-Steuerregister holen
EBEB  29 E5      AND #$E5        und alle Ausgänge löschen
EBED  8D 00 18   STA $1800       und Bus in Ausgangszustand setzen
EBF0  AD 55 02   LDA $0255       Flag für 'Kommando empfangen'
EBF3  F0 0A      BEQ $EBFF       gesetzt ?
EBF5  A9 00      LDA #$00        ja, Kommandoflag
EBF7  8D 55 02   STA $0255       löschen
EBFA  EA         NOP             [Leerschritte durch Modifizierung]
EBFB  EA         NOP             [des 1541 ROM]
EBFC  4C 1C A6   JMP $A61C       Befehl vom Rechner ausführen
-----------------------------------------------------------------------
```

[A628/EBF3/EC9B]

auf Befehl warten

```
EBFF     58         CLI           Bus-/Kontrollerroutine einschalten
EC00     A5 7C      LDA $7C       Flag für ATN empfangen
EC02     F0 03      BEQ $EC07     gesetzt ?
EC04     4C 94 A6   JMP $A7B3     ja, ATN-Kommando vom Bus annehmen
EC07[1]  58         CLI           Bus-/Kontrollerroutine wieder aktivieren
EC08     A9 0E      LDA #$0E      größte Sekundäradresse, die für
EC0A     85 72      STA $72       Dateien zur Verfügung gestellt wird
EC0C     A9 00      LDA #$00      Zähler für anstehende Jobs
EC0E     85 6F      STA $6F       für Laufwerk 0 und für
EC10     85 70      STA $70       Laufwerk 1 löschen
EC12[1]  A6 72      LDX $72       Sekundäradresse holen
EC14     BD 2B 02   LDA $022B,X   und entsprechenden Kanalstatus prüfen
EC17     C9 FF      CMP #$FF      mit Wert für 'frei' vergleichen
EC19     F0 10      BEQ $EC2B     ist Kanal frei ?
EC1B     29 3F      AND #$3F      nein, Nummer des zugeordneten internen
EC1D     85 82      STA $82       Kanals holen und merken
EC1F     20 93 DF   JSR $DF93     Puffernummer holen
EC22     AA         TAX           und merken
EC23     BD 5B 02   LDA $025B,X   Jobcode des Puffers feststellen
EC26     29 01      AND #$01      und letztes Laufwerk herstellen
EC28     AA         TAX           und merken
EC29     F6 6F      INC $6F,X     Zahl der Jobs erhöhen
EC2B[1]  C6 72      DEC $72       nächste Sekundäradresse
EC2D     10 E3      BPL $EC12     alle Kanäle berücksichtigt ?
EC2F     A0 04      LDY #$04      ja, Zahl der Puffer
EC31[1]  B9 00 00   LDA $0000,Y   Jobcode des Puffers holen
EC34     10 05      BPL $EC3B     ist Job in Arbeit ?
EC36     29 01      AND #$01      ja, Nummer des Laufwerks holen
EC38     AA         TAX           und merken
EC39     F6 6F      INC $6F,X     Zahl der Jobs erhöhen
EC3B[1]  88         DEY           nächsten Puffer anwählen
EC3C     10 F3      BPL $EC31     schon alle Jobs geprüft ?
EC3E     78         SEI           ja, Kontroller-/Busroutinen ausschalten
EC3F     AD 00 1C   LDA $1C00     Laufwerkssteuerregister holen
EC42     29 F7      AND #$F7      LED zurücksetzen und generierte
EC44     48         PHA           Maske für 'LED aus' merken
EC45     A5 7F      LDA $7F       aktuelles Laufwerk
EC47     85 86      STA $86       merken und
EC49     A9 00      LDA #$00      Laufwerk 0
```

```
EC4B   85 7F      STA $7F      anwählen
EC4D   A5 6F      LDA $6F      Zahl der Jobs für Laufwerk 0
EC4F   F0 0B      BEQ $EC5C    sind irgendwelche Jobs auszuführen ?
EC51   A5 1C      LDA $1C      ja, Flag für 'Diskette initialisieren'
EC53   F0 03      BEQ $EC58    soll Diskette initialisiert werden ?
EC55   20 13 D3   JSR $D313    Diskette initialisieren
EC58[1] 68        PLA          Maske für Laufwerkssteuerung wieder
EC59   09 08      ORA #$08     holen, LED anschalten und
EC5B   48         PHA          wieder merken
EC5C[1] E6 7F     INC $7F      Laufwerk 1 anwählen
EC5E   A5 70      LDA $70      Zähler für Jobs für laufwerk 1
EC60   F0 0B      BEQ $EC6D    sind für Laufwerk 1 Jobs auszuführen ?
EC62   A5 1D      LDA $1D      ja, Write-Protect-Flag für Laufwerk 1
EC64   F0 03      BEQ $EC69    hat ein Diskettenwechsel stattgefunden ?
EC66   20 13 D3   JSR $D313    alle Kanäle des Laufwerks abschließen
EC69[1] 68        PLA          Maske für Laufwerkssteuerung zurückholen
EC6A   09 00      ORA #$00     und LED an Laufwerk 1 setzen
EC6C   48         PHA          und wieder merken
EC6D[1] A5 86     LDA $86      aktuelles Laufwerk zurückholen
EC6F   85 7F      STA $7F      und übernehmen
EC71   68         PLA          Maske für Laufwerkssteuerung holen
-----------------------------------------------------------------------

LED-Fehlerblinken steuern
EC72   AE 6C 02   LDX $026C    Fehlerflag testen
EC75   F0 21      BEQ $EC98    gesetzt ?
EC77   AD 00 1C   LDA $1C00    ja, Steuerregister holen
EC7A   E0 80      CPX #$80     Zähler auf Blinkphase testen
EC7C   D0 03      BNE $EC81    Timer neu setzen ?
EC7E   4C 8B EC   JMP $EC8B    nein, weiter
EC81[1] AE 05 18  LDX $1805    High-Byte des Timer 1
EC84   30 12      BMI $EC98    ist Zähler abgelaufen
EC86   A2 A0      LDX #$A0     ja, Highbyte
EC88   8E 05 18   STX $1805    neu setzen
EC8B[1] CE 6C 02  DEC $026C    Blinkzähler erniedrigen
EC8E   D0 08      BNE $EC98    ist Zähler abgelaufen
EC90   4D 6D 02   EOR $026D    LED umschalten
EC93   A2 10      LDX #$10     Blinkzähler auf 0.4 / 0.2 Sekunden
EC95   8E 6C 02   STX $026C    Verzögerung setzen
EC98[3] 8D 00 1C  STA $1C00    LED ansteuern
EC9B   4C FF EB   JMP $EBFF    weiter blinken
-----------------------------------------------------------------------
```

[DAA7]

Directory für 'Load "$"' bereitstellen

Das Inhaltsverzeichnis wird dabei als Basicprogramm geladen

```
EC9E  A9 00     LDA #$00     Null als aktuelle
ECA0  85 83     STA $83      Sekundäradresse setzen (Load Kanal)
ECA2  A9 01     LDA #$01     einen Lesekanal suchen und
ECA4  20 E2 D1  JSR $D1E2    dazugehöhriger Puffer festlegen
ECA7  A9 00     LDA #$00     Zeiger auf Position
ECA9  20 C8 D4  JSR $D4C8    im Puffer auf Null zurücksetzen
ECAC  A6 82     LDX $82      Nummer des gefundenen Kanals
ECAE  A9 00     LDA #$00     Zeiger auf Ende des gültigen Puffer-
ECB0  9D 44 02  STA $0244,X  eintrags löschen
ECB3  20 93 DF  JSR $DF93    Nummer des ausgewählten Puffers holen
ECB6  AA        TAX          und merken
ECB7  A5 7F     LDA $7F      aktuelles Laufwerk holen und in
ECB9  9D 5B 02  STA $025B,X  Laufwerkstabelle dem Puffer zuordnen
ECBC  A9 01     LDA #$01     Startadresse des 'fiktiven'
ECBE  20 F1 CF  JSR $CFF1    Basicprogramms ($0401)
ECC1  A9 04     LDA #$04     aktuellen Puffer
ECC3  20 F1 CF  JSR $CFF1    schreiben
ECC6  A9 01     LDA #$01     zwei Füllbytes als Platzhalter
ECC8  20 F1 CF  JSR $CFF1    für die Verkettungszeiger der
ECCB  20 F1 CF  JSR $CFF1    Basiczeilen in den Puffer schreiben
ECCE  AD 72 02  LDA $0272    Nummer des Laufwerks holen
ECD1  20 F1 CF  JSR $CFF1    und in Puffer als Low-Byte der Zeilen-
ECD4  A9 00     LDA #$00     nummer übergeben; das Highbyte
ECD6  20 F1 CF  JSR $CFF1    wird auf Null gesetzt
ECD9  20 59 ED  JSR $ED59    Diskettennamen in Puffer übertragen
ECDC  20 93 DF  JSR $DF93    Nummer des aktuellen
ECDF  0A        ASL A        Puffers holen, verdoppeln
ECE0  AA        TAX          und Zeiger auf aktuelle Position im
ECE1  D6 99     DEC $99,X    Puffer um zwei Zeichen
ECE3  D6 99     DEC $99,X    erniedrigen
ECE5  A9 00     LDA #$00     Endekennzeichen der Basiczeile
ECE7  20 F1 CF  JSR $CFF1    in Puffer speichern
ECEA  A9 01     LDA #$01     Zwei Bytes als Platzhalter
ECEC  20 F1 CF  JSR $CFF1    für die Zeilenverkettung der
ECEF  20 F1 CF  JSR $CFF1    Basiczeilen setzen
ECF2  20 CE C6  JSR $C6CE    Eintrag aus Directory lesen
ECF5  90 2C     BCC $ED23    alle Einträge bearbeitet ?
ECF7  AD 72 02  LDA $0272    nein, Zahl der belegten Blöcke (Low)
```

```
ECFA  20 F1 CF   JSR $CFF1      als Low-Byte der Basiczeilennummer und
ECFD  AD 73 02   LDA $0273      High-Byte der Blockanzahl als High-Byte
ED00  20 F1 CF   JSR $CFF1      der Basiczeilennummer in Puffer
ED03  20 59 ED   JSR $ED59      Directoryeintrag in Puffer kopieren
ED06  A9 00      LDA #$00       Kennzeichen für 'Basiczeile zu Ende'
ED08  20 F1 CF   JSR $CFF1      in Puffer setzen
ED0B  D0 DD      BNE $ECEA      ist der Puffer voll ?
ED0D  20 93 DF   JSR $DF93      nein, Nummer des aktuellen Puffers holen
ED10  0A         ASL A          Nummer verdoppeln
ED11  AA         TAX            und den Zeiger auf die aktuelle
ED12  A9 00      LDA #$00       Position im entsprechenden
ED14  95 99      STA $99,X      Puffer wieder zurücksetzen
ED16  A9 88      LDA #$88       Flagwert für 'Directory nicht im Puffer'
ED18  A4 82      LDY $82        Nummer des Kanals holen
ED1A  8D 54 02   STA $0254      Flag setzen
ED1D  99 F2 00   STA $00F2,Y    Kanalsstatus auf Lesen schalten
ED20  A5 85      LDA $85        aktuelles Datenbyte holen
ED22  60         RTS            zurück zur aufrufenden Routine
-----------------------------------------------------------------------
[ECF5]
Ausgabe des Directory abschließen
ED23  AD 72 02   LDA $0272      Zahl der freien Blöcke in
ED26  20 F1 CF   JSR $CFF1      $0272/$0273 als
ED29  AD 73 02   LDA $0273      Basiczeilennummer nehmen und in
ED2C  20 F1 CF   JSR $CFF1      Puffer schreiben
ED2F  20 59 ED   JSR $ED59      schreibt 'Blocks Free' in Puffer
ED32  20 93 DF   JSR $DF93      Nummer des aktuellen Puffers holen
ED35  0A         ASL A          Nummer verdoppeln
ED36  AA         TAX            Zeiger auf die aktuelle
ED37  D6 99      DEC $99,X      Zeichenposition im Puffer um zwei
ED39  D6 99      DEC $99,X      Byte vorsetzen
ED3B  A9 00      LDA #$00       Kennzeichen für Basiczeilenende und
ED3D  20 F1 CF   JSR $CFF1      zwei leere Verkettungsbytes als
ED40  20 F1 CF   JSR $CFF1      Kennzeichen für das Programmende
ED43  20 F1 CF   JSR $CFF1      in aktuellen Puffer übernehmen
ED46  20 93 DF   JSR $DF93      aktuelle Puffernummer holen
ED49  0A         ASL A          Nummer verdoppeln
ED4A  A8         TAY            Zahl der im Puffer noch
ED4B  B9 99 00   LDA $0099,Y    gültigen Bytes holen
ED4E  A6 82      LDX $82        und als Zeiger auf das Ende für
ED50  9D 44 02   STA $0244,X    die, aus dem Puffer  zu übertragenden
```

```
ED53  DE 44 02   DEC $0244,X      Bytes setzen
ED56  4C 0D ED   JMP $ED0D        Ende
--------------------------------------------------------------------
[ECD9/ED03/ED2F]
Directoryeintrag in aktuellen Puffer kopieren
ED59  A0 00      LDY #$00         Pufferzeiger initialisieren
ED5B  B9 B1 02   LDA $02B1,Y      und ein Zeichen aus dem Directorypuffer
ED5E  20 F1 CF   JSR $CFF1        holen und in aktuellen Puffer übertragen
ED61  C8         INY              Zeiger auf nächstes Zeichen setzen
ED62  C0 1B      CPY #$1B         Zahl der Zeichen pro Eintrag
ED64  D0 F5      BNE $ED5B        alle Zeichen kopiert ?
ED66  60         RTS              ja, zurück zur aufrufenden Routine
--------------------------------------------------------------------
[D40E]
Byte aus Directory holen
ED67  20 37 D1   JSR $D137        Byte aus Datei holen
ED6A  F0 01      BEQ $ED6D        Ende der Datei erreicht ?
ED6C  60         RTS              nein, zurück zur aufrufenden Routine
ED6D  85 85      STA $85          letztes Datenbyte merken
ED6F  A4 82      LDY $82          Nummer des Kanals holen
ED71  B9 44 02   LDA $0244,Y      Zahl der zu übertragenden Bytes holen
ED74  F0 08      BEQ $ED7E        keine Daten mehr ?
ED76  A9 80      LDA #$80         nein, Kanalstatus auf 'Lesen/EOI'
ED78  99 F2 00   STA $00F2,Y      setzen
ED7B  A5 85      LDA $85          und letztes Datenbyte wieder holen
ED7D  60         RTS              zurück zur aufrufenden Routine
ED7E  48         PHA              Zahl der Datenbytes holen
ED7F  20 EA EC   JSR $ECEA        Directoryzeile erzeugen
ED82  68         PLA              letztes Datenbyte wieder holen
ED83  60         RTS              zurück zur aufrufenden Routine
--------------------------------------------------------------------
[Einsprung über Routine C146]
Routine für Validate Befehl
ED84  20 D1 C1   JSR $C1D1        Laufwerknummer aus Befehlsstring holen
ED87  20 42 D0   JSR $D042        Diskette initialisieren
ED8A  A9 40      LDA #$40         Flag für 'BAM ungültig'
ED8C  8D F9 02   STA $02F9        setzen
ED8F  20 C7 A7   JSR $A7C7        neue BAM erzeugen
ED92  A9 00      LDA #$00         Zeiger auf Directoryeintrag
ED94  8D 92 02   STA $0292        löschen; Flag für Suche setzen
ED97  20 AC C5   JSR $C5AC        Dateieintrag in Directory suchen
```

```
ED9A  D0 3D      BNE $EDD9       Eintrag gefunden ?
ED9C  A9 00      LDA #$00        nein, Nummer des Sektors
ED9E  85 81      STA $81         auf Null setzen
EDA0  AD 85 FE   LDA $FE85       Nummer der Directoryspur (18)
EDA3  85 80      STA $80         holen und setzen
EDA5  20 E5 ED   JSR $EDE5       Directoryspur in BAM belegen
EDA8  A9 00      LDA #$00        Flag für 'BAM ungültig'
EDAA  8D F9 02   STA $02F9       löschen
EDAD  20 FF EE   JSR $EEFF       BAM auf Diskette zurückschreiben
EDB0  4C 94 C1   JMP $C194       'OK' Meldung ausgeben und Befehl beenden
-----------------------------------------------------------------------

[EDDD]
Alle Blöcke einer Datei in der BAM belegen
EDB3  C8         INY             Zeiger in Directorypuffer auf Nummer
EDB4  B1 94      LDA ($94),Y     der Spur des ersten Dateiblocks setzen
EDB6  48         PHA             und Spurnummer holen und merken
EDB7  C8         INY             Pufferzeiger auf nächstes Zeichen
EDB8  B1 94      LDA ($94),Y     Sektornummer des ersten Dateiblocks
EDBA  48         PHA             holen und merken
EDBB  A0 13      LDY #$13        Pufferzeiger auf Position des Side-
EDBD  B1 94      LDA ($94),Y     Sektor-Zeigers setzen und Spur holen
EDBF  F0 0A      BEQ $EDCB       Side-Sektorblock verfügbar ?
EDC1  85 80      STA $80         ja, Spurnummer des ersten Side-Sektors
EDC3  C8         INY             merken und Sektornummer aus
EDC4  B1 94      LDA ($94),Y     Directorypuffer holen
EDC6  85 81      STA $81         und merken
EDC8  20 E5 ED   JSR $EDE5       Side-Sektor Blöcke lesen und belegen
EDCB  68         PLA             Sektornummer wieder holen
EDCC  85 81      STA $81         und merken
EDCE  68         PLA             Spurnummer wieder herstellen
EDCF  85 80      STA $80         und merken
EDD1  20 E5 ED   JSR $EDE5       Datenblöcke lesen und in BAM belegen
EDD4  20 04 C6   JSR $C604       nächsten gültigen Dateieintrag holen
EDD7  F0 C3      BEQ $ED9C       alle Dateien überprüft ?
EDD9  A0 00      LDY #$00        Pufferzeiger auf erstes Zeichen setzen
EDDB  B1 94      LDA ($94),Y     Kennzeichen für Dateityp holen
EDDD  30 D4      BMI $EDB3       wurde Datei ordnungsgemäß geschlossen ?
EDDF  20 B6 C8   JSR $C8B6       nein, Datei löschen
EDE2  4C D4 ED   JMP $EDD4       mit nächstem Eintrag weitermachen
-----------------------------------------------------------------------
```

```
[EDA5/EDC8/EDD1]
Alle Blöcke einer Datei verfolgen
EDE5  20 5F D5   JSR $D55F      Spur- und Sektornummer überprüfen
EDE8  20 90 EF   JSR $EF90      aktuellen Block in BAM belegen
EDEB  20 75 D4   JSR $D475      Block in Puffer lesen
EDEE  A9 00      LDA #$00       Pufferzeiger auf Beginn des
EDF0  20 C8 D4   JSR $D4C8      Datenblocks setzen
EDF3  20 37 D1   JSR $D137      erstes Byte aus Datenblock holen und als
EDF6  85 80      STA $80        Spurnummer des nächsten Blocks merken
EDF8  20 37 D1   JSR $D137      zweites Byte aus Datenblock holen
EDFB  85 81      STA $81        und als dazugehörigen Sektor speichern
EDFD  A5 80      LDA $80        Spurnummer auf Kennzeichen für Dateiende
EDFF  D0 03      BNE $EE04      prüfen; letzter Block der Datei ?
EE01  4C 27 D2   JMP $D227      ja, Kanal schließen und beenden
EE04  20 90 EF   JSR $EF90      Block in BAM als belegt kennzeichnen
EE07  20 4D D4   JSR $D44D      nächsten Datenblock lesen
EE0A  4C EE ED   JMP $EDEE      und Prüfung fortführen
-----------------------------------------------------------------------
[Einsprung über Routine C146/EE43:A776]
Routine für New Befehl (Diskette formatieren)
EE0D  20 12 C3   JSR $C312      Laufwerksnummern aus Befehl holen
EE10  A5 E2      LDA $E2        Nummer holen
EE12  10 05      BPL $EE19      Nummer in Ordnung ?
EE14  A9 33      LDA #$33       nein, Fehlermeldung
EE16  4C C8 C1   JMP $C1C8      '33 Syntax Error' ausgeben
EE19  29 01      AND #$01       Nummer des Laufwerks herstellen
EE1B  85 7F      STA $7F        und als aktuelles Laufwerk merken
EE1D  20 9C FF   JSR $FF9C      Laufwerksstatus setzen und LED an
EE20  A5 7F      LDA $7F        Nummer des aktuellen Laufwerks holen
EE22  0A         ASL A          und verdoppeln
EE23  AA         TAX            (da 2-Byte-Tabelle
EE24  AC 7B 02   LDY $027B      Zeiger auf Position der ID im Befehl
EE27  CC 74 02   CPY $0274      mit Länge des Befehlsstring vergleichen
EE2A  F0 1A      BEQ $EE46      ist eine neue ID angegeben ?
EE2C  B9 00 02   LDA $0200,Y    ja, erstes Zeichen der ID aus Eingabe-
EE2F  95 12      STA $12,X      puffer holen und übernehmen
EE31  B9 01 02   LDA $0201,Y    zweites Zeichen der ID holen
EE34  95 13      STA $13,X      und merken
EE36  20 07 D3   JSR $D307      alle Kanäle abschließen
EE39  A9 01      LDA #$01       erste Spur, die formatiert werden soll
EE3B  85 80      STA $80        setzen
```

```
EE3D   20 2F FF   JSR $FF2F      Diskette formatieren
EE40   4C 64 A7   JMP $A764      Puffer für BAM löschen
EE43   4C 56 EE   JMP $EE56      Sektor 18,0 erzeugen
EE46 1 20 42 D0   JSR $D042      Diskette initialisieren
EE49   A6 7F      LDX $7F        aktuelles Laufwerk holen und
EE4B   BD 01 01   LDA $0101,X    dazugehöhriges gelesenes Formatkenn-
EE4E   CD D5 FE   CMP $FED5      zeichen feststellen
EE51   F0 03      BEQ $EE56      richtiges Format ?
EE53   4C 72 D5   JMP $D572      nein, Einschaltmeldung ausgeben
EE56 2 20 C7 A7   JSR $A7C7      neue BAM errzeugen
EE59   A5 F9      LDA $F9        Nummer des aktuellen Puffers
EE5B   A8         TAY            Nummer verdoppeln
EE5C   0A         ASL A          (da Pufferzeiger aus
EE5D   AA         TAX            2-Byte-Werten bestehen)
EE5E   AD 88 FE   LDA $FE88      Position des Disknamens in Sektor 18,0
EE61   95 99      STA $99,X      holen und in Pufferzeiger übernehmen
EE63   AE 7A 02   LDX $027A      Puffernummer holen
EE66   A9 1B      LDA #$1B       Länge des Diskettennamens
EE68   20 6E C6   JSR $C66E      Diskettennamen in BAM-Puffer kopieren
EE6B   A0 12      LDY #$12       Zeiger in Diskettenname
EE6D   A6 7F      LDX $7F        aktuelles Laufwerk holen
EE6F   AD D5 FE   LDA $FED5      Kennzeichen für 1541/1571-Format holen
EE72   9D 01 01   STA $0101,X    und merken
EE75   8A         TXA            Nummer des Laufwerks
EE76   0A         ASL A          holen und
EE77   AA         TAX            verdoppeln
EE78   B5 12      LDA $12,X      ID des Laufwerks holen und in
EE7A   91 94      STA ($94),Y    Puffer übernehmen
EE7C   C8         INY            Pufferzeiger auf nächstes Byte setzen
EE7D   B5 13      LDA $13,X      zweites Zeichen der ID holen
EE7F   91 94      STA ($94),Y    und in Puffer übernehmen
EE81   C8         INY            Pufferzeiger zwei Zeichen
EE82   C8         INY            weiterrücken
EE83   A9 32      LDA #$32       '2A' als
EE85   91 94      STA ($94),Y    Kennzeichen für Format
EE87   C8         INY            in Directoryzeile
EE88   AD D5 FE   LDA $FED5      mit Diskettennamen und ID
EE8B   91 94      STA ($94),Y    übernehmen
EE8D   A0 02      LDY #$02       Spurnummer
EE8F   91 6D      STA ($6D),Y    in BAM schreiben
EE91   AD 85 FE   LDA $FE85      Nummer der Spur für Directory
```

```
EE94  85 80     STA $80       setzen
EE96  20 93 EF  JSR $EF93     BAM-Block in BAM als belegt setzen
EE99  A9 01     LDA #$01      Nummer des ersten Directoryblocks
EE9B  85 81     STA $81       festlegen
EE9D  20 93 EF  JSR $EF93     Directoryblock in BAM legen
EEA0  20 FF EE  JSR $EEFF     neu erstellte BAM auf Diskette schreiben
EEA3  20 05 F0  JSR $F005     Puffer für BAM löschen
EEA6  A0 01     LDY #$01      Pufferzeiger auf zweites Zeichen setzen
EEA8  A9 FF     LDA #$FF      Zahl der gültigen Bytes des Puffers
EEAA  91 6D     STA ($6D),Y   in Directoryblock schreiben
EEAC  20 64 D4  JSR $D464     Directoryblock 18,1 schreiben
EEAF  C6 81     DEC $81       aktuelle Sektornummer auf Null
EEB1  20 42 D0  JSR $D042     und Sektor lesen
EEB4  4C 94 C1  JMP $C194     'Ok' Meldung bereitstellen
--------------------------------------------------------------------------

[A6C4/A708]
Neue 1541-BAM anlegen
EEB7  20 D1 F0  JSR $F0D1     Puffer für BAM löschen
EEBA  A0 00     LDY #$00      Pufferzeiger initialisieren
EEBC  A9 12     LDA #$12      Zeiger auf Spurnummer des nächsten
EEBE  91 6D     STA ($6D),Y   Blocks auf Spur 18 richten
EEC0  C8        INY           Pufferzeiger auf Sektornummer setzen
EEC1  98        TYA           (1)
EEC2  91 6D     STA ($6D),Y   Sektornummer 1 übernehmen
EEC4  C8        INY           aktuellen Pufferzeiger
EEC5  C8        INY           um drei Zeichenpositionen
EEC6  C8        INY           weiterrücken
EEC7  A9 00     LDA #$00      Zwischenspeicher zum
EEC9  85 6F     STA $6F       Erstellen der Liste
EECB  85 70     STA $70       der belegten Blöcke
EECD  85 71     STA $71       löschen
EECF  98        TYA           Nummer der Spur
EED0  4A        LSR A         festlegen, für die die
EED1  4A        LSR A         Blockbelegung ermittelt wird
EED2  20 4B F2  JSR $F24B     maximale Zahl der Sektoren feststellen
EED5  91 6D     STA ($6D),Y   und in BAM übernehmen
EED7  C8        INY           Pufferzeiger auf nächstes Byte
EED8  AA        TAX           Zähler für Zahl der Sektoren setzen
EED9  38        SEC           Bitflag für 'Sektor belegt' setzen
EEDA  26 6F     ROL $6F       Bit in 24-Bit-Zwischenspeicher
EEDC  26 70     ROL $70       der belegten Blöcke der Spur
```

```
EEDE    26 71       ROL $71         ablegen
EEE0    CA          DEX             nächsten Sektor in BAM setzen
EEE1    D0 F6       BNE $EED9       schon alle Sektoren der Spur ?
EEE3[1] B5 6F       LDA $6F,X       ja, Inhalt des Zwischenspeichers
EEE5    91 6D       STA ($6D),Y     in Puffer für BAM schreiben
EEE7    C8          INY             Pufferzeiger auf nächstes Byte
EEE8    E8          INX             Zähler für Zahl der Zwischenspeicher
EEE9    E0 03       CPX #$03        mit drei vergleichen
EEEB    90 F6       BCC $EEE3       alle Zwischenspeicherbytes kopiert ?
EEED    C0 90       CPY #$90        ja, Pufferzeiger mit $90 vergleichen
EEEF    90 D6       BCC $EEC7       BAM-Bits aller Spuren festgelegt ?
EEF1    4C 75 D0    JMP $D075       ja, 'Blocks Free' berechnen
---------------------------------------------------------------------------
[C8A7/DB26/DD87/E433/EEFF:EDAD,EEA0]
BAM korregieren und auf Diskette zurückschreiben
EEF4    20 93 DF    JSR $DF93       aktuelle Puffernummer holen
EEF7    AA          TAX             Nummer des dazugehörigen
EEF8    BD 5B 02    LDA $025B,X     Jobcodes holen
EEFB    29 01       AND #$01        daraus Laufwerksnummer feststellen
EEFD    85 7F       STA $7F         und als aktuelles Laufwerk merken
EEFF[2] A4 7F       LDY $7F         Zum Laufwerk passendes Flag
EF01    B9 51 02    LDA $0251,Y     für 'BAM ungültig' holen
EF04    D0 01       BNE $EF07       muss eine neue BAM erzeugt werden
EF06    60          RTS             nein, zurück zur aufrufenden Routine
EF07[1] A9 00       LDA #$00        Flag für 'BAM ungültig'
EF09    99 51 02    STA $0251,Y     löschen
EF0C    20 3A EF    JSR $EF3A       BAM in Puffer holen und Zeiger setzen
EF0F    A5 7F       LDA $7F         aktuelles Laufwerk holen
EF11    0A          ASL A           und Nummer verdoppeln
EF12    48          PHA             Wert merken
EF13    20 A5 F0    JSR $F0A5       Zwischenspeicher in BAM kopieren
EF16    68          PLA             Lauwerkszeiger wieder
EF17    18          CLC             holen
EF18    69 01       ADC #$01        nächsten ZS anwählen
EF1A    20 A5 F0    JSR $F0A5       Zwischenspeicher in BAM übertragen
EF1D    A5 80       LDA $80         Nummer der aktuellen Spur
EF1F    48          PHA             retten
EF20    A9 01       LDA #$01        Nummer auf Spur 1
EF22    85 80       STA $80         setzen
EF24[1] 0A          ASL A           mal vier nehmen
EF25    0A          ASL A           (da 4 BAM-Bytes pro Spur)
```

```
EF26  85 6D     STA $6D        Position der Spurbytes merken
EF28  20 37 A9  JSR $A937      Zahl der freien Blocks überprüfen
EF2B  E6 80     INC $80        Zeiger auf nächste Spur setzen
EF2D  A5 80     LDA $80        und holen
EF2F  CD AC 02  CMP $02AC      Nummer der letzten Spur +1
EF32  90 F0     BCC $EF24      letzte Spur erreicht ?
EF34  68        PLA            alte Spurnummer
EF35  85 80     STA $80        wieder einrichten
EF37  4C 8D A5  JMP $A58D      BAM auf Diskette schreiben
----------------------------------------------------------------------
[A5AA/A738/D075/EF0C]
BAM lesen und Pufferzeiger setzen
EF3A  20 0F F1  JSR $F10F      Kanalnummer für 'BAM lesen'
EF3D  AA        TAX            holen und merken
EF3E  20 DF F0  JSR $F0DF      dazugehörigen Puffer belegen
EF41  A6 F9     LDX $F9        Nummer des Puffers holen
EF43  BD E0 FE  LDA $FEE0,X    und Speicheradresse des Puffers
EF46  85 6E     STA $6E        feststellen
EF48  A9 00     LDA #$00       Speicheradresse dann in
EF4A  85 6D     STA $6D        Zeiger $6D/$6E übernehmen
EF4C  60        RTS            zurück zur aufrufenden Routine
----------------------------------------------------------------------
[C814/D33B]
Anzahl der 'Blocks Free' holen
EF4D  A6 7F     LDX $7F        Nummer des aktuellen Laufwerks
EF4F  BD FA 02  LDA $02FA,X    Zahl der freien Blöcke (Low-Byte)
EF52  8D 72 02  STA $0272      übernehmen
EF55  BD FC 02  LDA $02FC,X    Zahl der freien Blöcke (High-Byte)
EF58  8D 73 02  STA $0273      übernehmen
EF5B  60        RTS            zurück zur aufrufenden Routine
----------------------------------------------------------------------
EF5C  20 F1 EF  JSR $EFF1      unbenutzter Programmrest aus 1541 DOS
----------------------------------------------------------------------
[C87D/C8AD/CCF8/EF62:A848/EF7F:A86F]
Sektor freigeben
EF5F  4C 27 A7  JMP $A727      Sektor in 1571-BAM freigeben
EF62  38        SEC            Flag für 'Sektor schon frei'
EF63  D0 22     BNE $EF87      ist der Block schon freigegeben ?
EF65  B1 6D     LDA ($6D),Y    nein, Bitmuster der Spur holen
EF67  1D E9 EF  ORA $EFE9,X    Sektor freigeben (Bit=1)
EF6A  91 6D     STA ($6D),Y    und wieder in BAM übernehmen
```

```
EF6C  20 88 EF   JSR $EF88      Flag für 'BAM schreiben' setzen
EF6F  A4 6F      LDY $6F        Zeiger auf aktuelles BAM-Byte
EF71  18         CLC            Flag für 'Sektor freigegeben'
EF72  B1 6D      LDA ($6D),Y    Zahl der freien Blocks der Spur
EF74  69 01      ADC #$01       erhöhen
EF76  91 6D      STA ($6D),Y    und wieder setzen
EF78  A5 80      LDA $80        Nummer der bearbeiteten Spur holen und
EF7A  CD 85 FE   CMP $FE85      mit Nummer für Directoryspur vergleichen
EF7D  F0 3B      BEQ $EFBA      identisch ?
EF7F[1] FE FA 02 INC $02FA,X    nein, Zahl der Blocks der Diskette +1
EF82  D0 03      BNE $EF87      ist ein Übertrag entstanden
EF84  FE FC 02   INC $02FC,X    Übertrag berücksichtigen
EF87[2] 60       RTS            zurück zur aufrufenden Routine
-----------------------------------------------------------------
[A85F/A895/EF6C/EF9F]
Flag für 'BAM auf Diskette ungültig' setzen
EF88  A6 7F      LDX $7F        aktuelles Laufwerk feststellen
EF8A  A9 01      LDA #$01       und dazugehöriges Flag für
EF8C  9D 51 02   STA $0251,X    'BAM ungültig' setzen
EF8F  60         RTS            zurück zur aufrufenden Routine
-----------------------------------------------------------------
[CD13/EDE8/EE04/F19A/F1F2/EFB2:A8A5]
Sektor in der BAM belegen
EF90  20 F1 EF   JSR $EFF1      letzte BAM-Änderung auf Disk schreiben
EF93  4C 74 A8   JMP $A874      Sektor in 1571 BAM belegen
EF96  F0 36      BEQ $EFCE      ist Sektor schon belegt ?
EF98  B1 6D      LDA ($6D),Y    nein, Byte mit Bitmuster für belegte
                                Blocks
EF9A  5D E9 EF   EOR $EFE9,X    holen und Sektor belegen (Bit=0)
EF9D  91 6D      STA ($6D),Y    BAM-Byte wieder merken
EF9F  20 88 EF   JSR $EF88      Flag für 'BAM schreiben' setzen
EFA2  A4 6F      LDY $6F        Zeiger auf aktuelles BAM-Byte
EFA4  B1 6D      LDA ($6D),Y    Zahl der freien Sektoren
EFA6  38         SEC            der Spur
EFA7  E9 01      SBC #$01       erniedrigen und wieder
EFA9  91 6D      STA ($6D),Y    in BAM schreiben
EFAB  A5 80      LDA $80        Nummer der bearbeiteten Spur mit
EFAD  CD 85 FE   CMP $FE85      Nummer der Directoryspur vergleichen
EFB0  F0 0B      BEQ $EFBD      identisch ?
EFB2[1] BD FA 02 LDA $02FA,X    Zahl der freien Blocks der Diskette
EFB5  D0 03      BNE $EFBA      entsteht ein Borgen ?
```

```
EFB7   DE FC 02   DEC $02FC,X    ja, Highbyte des Zählers erniedrigen
EFBA[2] DE FA 02  DEC $02FA,X    Zahl der freien Blocks -1
EFBD[1] BD FC 02  LDA $02FC,X    Zahl der freien Blöcke (High-Byte)
EFC0   D0 0C      BNE $EFCE      kleiner als 255 ?
EFC2   BD FA 02   LDA $02FA,X    ja, Zahl der freien Blöcke (Low-Byte)
EFC5   C9 03      CMP #$03       mit drei vergleichen
EFC7   B0 05      BCS $EFCE      weniger als drei Blöcke frei ?
EFC9   A9 72      LDA #$72       ja, Fehlermeldung
EFCB   20 C7 E6   JSR $E6C7      '72 Disk Full' ausgeben
EFCE[3] 60        RTS            zurück zur aufrufenden Routine
--------------------------------------------------------------------------
[A845/A87B]
BAM-Pufferzeiger auf Bit für aktuellen Sektor setzen und Bit holen
EFCF   20 11 F0   JSR $F011      Zeiger auf Beginn der Bitmuster für
EFD2   98         TYA            Spur errechnen
EFD3   85 6F      STA $6F        und merken
EFD5   A5 81      LDA $81        Nummer des zu bearbeitenden
EFD7   4A         LSR A          Sektors holen und
EFD8   4A         LSR A          durch acht teilen (8 Bits pro Byte)
EFD9   4A         LSR A          um Nummer des BAM-Bytes zu erhalten
EFDA   38         SEC            dazu eins und die
EFDB   65 6F      ADC $6F        ermittelte Zeigerposition addieren
EFDD   A8         TAY            und merken
EFDE   A5 81      LDA $81        aktuelle Sektornummer holen
EFE0   29 07      AND #$07       Nummer des Bits im BAM-Byte errechnen
EFE2   AA         TAX            und merken
EFE3   B1 6D      LDA ($6D),Y    Byte aus der BAM holen und
EFE5   3D E9 EF   AND $EFE9,X    Bit des Sektors isolieren
EFE8   60         RTS            zurück zur aufrufenden Routine
--------------------------------------------------------------------------
[A4FF/A56E/A859/A88F/D2BF/D2CB/EF67/EF9A/EFE5/F22F]
EFE9   01 02 04 08 10 20 40 80   Masken um BAM-Bits zu isolieren
--------------------------------------------------------------------------
[EF5C/EF90]
BAM auf Diskette schreiben
EFF1   A9 FF      LDA #$FF       Wert für 'BAM ungültig' Flag setzen
EFF3   2C F9 02   BIT $02F9      Zustand des Flags prüfen
EFF6   F0 0C      BEQ $F004      gleich Null ?
EFF8   10 0A      BPL $F004      nein, Bit 7 gelöscht ?
EFFA   70 08      BVS $F004      nein, Bit 6 gelöscht ?
EFFC   A9 00      LDA #$00       ja, Flag für 'BAM ungültig, neue BAM
```

```
EFFE     8D F9 02   STA $02F9       schreiben' zurücksetzen
F001     4C 0D A5   JMP $A58D       BAM auf Diskette zurückschreiben
F004[3]  60         RTS             zurück zur aufrufenden Routine
----------------------------------------------------------------------
[A764/BF3C/EEA3/F008:A4AD,A74C]
Puffer für BAM löschen
F005     20 25 A6   JSR $A625       Zeiger auf BAM-Puffer setzen
F008[2]  A0 00      LDY #$00        Zeiger auf Position im Puffer löschen
F00A     98         TYA             Wert mit dem Puffer gefüllt wird (0)
F00B[1]  91 6D      STA ($6D),Y     in Puffer schreiben
F00D     C8         INY             Zeiger auf nächstes Byte setzen
F00E     D0 FB      BNE $F00B       war das letztes Byte des Puffers ?
F010     60         RTS             ja, zurück zur aufrufenden Routine
----------------------------------------------------------------------
[A8B0/A90C/A925/EFCF/F130/F01B:AA2A]
Zeiger für BAM-Zwischenspeicher setzen
F011     A5 6F      LDA $6F         Zeropageadressen $6F/$70
F013     48         PHA             werden als Zwischenspeicher
F014     A5 70      LDA $70         für die Routine verwendet und
F016     48         PHA             deshalb gerettet
F017     A6 7F      LDX $7F         aktuelle Laufwerksnummer
F019     B5 FF      LDA $FF,X       und dazugehörigen Laufwerksstatus holen
F01B[1]  F0 05      BEQ $F022       ist das Laufwerk bereit ?
F01D     A9 74      LDA #$74        nein, Fehlermeldung
F01F     20 48 E6   JSR $E648       '74 Drive Not Ready' ausgeben
F022[1]  20 0F F1   JSR $F10F       Puffer- und Kanalnummer feststellen
F025     85 6F      STA $6F         Kanalnummer übernehmen
F027     8A         TXA             Puffernummer
F028     0A         ASL A           verdoppeln
F029     85 70      STA $70         und übernehmen
F02B     AA         TAX             Wert merken
F02C     A5 80      LDA $80         Nummer der aktuellen Spur
F02E     DD 9D 02   CMP $029D,X     auf Spurdaten in Zwischenspeicher testen
F031     F0 0B      BEQ $F03E       identisch ?
F033     E8         INX             nein, auf nächsten Zwischenspeicher
F034     86 70      STX $70         wechseln
F036     DD 9D 02   CMP $029D,X     mit Spur des ZS vergleichen
F039     F0 03      BEQ $F03E       sind Daten hier enthalten ?
F03B     20 5B F0   JSR $F05B       nein, Spurdaten in Speicher holen
F03E[2]  A5 70      LDA $70         Zeiger auf Zwischenspeicher
F040     A6 7F      LDX $7F         aktuelles Laufwerk
```

```
F042  9D 9B 02   STA $029B,X    Pufferzeiger merken
F045  0A         ASL A          Wert vervierfachen
F046  0A         ASL A          (da 4 Bytes pro Eintrag)
F047  18         CLC            neuen BAM Zeiger
F048  69 A1      ADC #$A1       auf Position
F04A  85 6D      STA $6D        des Zwischenspeichers
F04C  A9 02      LDA #$02       einrichten
F04E  69 00      ADC #$00       High-Byte des Zeigers
F050  85 6E      STA $6E        setzen
F052  A0 00      LDY #$00       Zeiger auf aktuelles Byte
F054  68         PLA            Zeropageadressen
F055  85 70      STA $70        $6F und $70 wieder
F057  68         PLA            auf den alten Wert
F058  85 6F      STA $6F        zurücksetzen
F05A  60         RTS            zurück zur aufrufenden Routine
-----------------------------------------------------------------
[F03B]
BAM-Bytes aus BAM in Zwischenspeicher kopieren
F05B  A6 6F      LDX $6F        Nummer des Kanals holen
F05D  20 DF F0   JSR $F0DF      BAM von Diskette lesen
F060  A5 7F      LDA $7F        aktuelle Laufwerksnummer
F062  AA         TAX            holen und merken
F063  0A         ASL A          Nummer verdoppeln (da 2 Laufwerke)
F064  1D 9B 02   ORA $029B,X    alten ZS Nummer einrechnen
F067  49 01      EOR #$01       auf anderen Zwischenspeicher
F069  29 03      AND #$03       umschalten
F06B  85 70      STA $70        und neuen Zeiger merken
F06D  20 A5 F0   JSR $F0A5      aktuellen ZS-Inhalt wieder in BAM zurück
F070  A5 F9      LDA $F9        Nummer des aktuellen Puffers
F072  0A         ASL A          verdoppeln (da Zeiger 2-Byte-Werte
F073  AA         TAX            sind) und merken
F074  A5 80      LDA $80        Nummer der aktuellen Spur
F076  0A         ASL A          vervierfachen
F077  0A         ASL A          (da 4 BAM-Bytes pro Spur)
F078  95 99      STA $99,X      Wert in Puffer schreiben
F07A  A5 70      LDA $70        Zeiger auf aktuellen ZS
F07C  0A         ASL A          vervierfachen
F07D  0A         ASL A          (da 4 verschiedene ZS)
F07E  A8         TAY            und setzen
F07F  A1 99      LDA ($99,X)    Byte aus BAM holen
F081  99 A1 02   STA $02A1,Y    und in Zwischenspeicher schreiben
```

```
F084  A9 00      LDA #$00      Werte in BAM
F086  81 99      STA ($99,X)   löschen
F088  F6 99      INC $99,X     Zeiger auf nächstes Byte
F08A  C8         INY           Zeiger auf nächstes ZS Zeichen
F08B  98         TYA           Zeiger auf
F08C  29 03      AND #$03      Wert 4 prüfen
F08E  D0 EF      BNE $F07F     alle Bytes in ZS kopiert ?
F090  A6 70      LDX $70       ja, Nummer des aktuellen ZS holen
F092  A5 80      LDA $80       Nummer der dazugehörigen Spur
F094  9D 9D 02   STA $029D,X   vermerken
F097  AD F9 02   LDA $02F9     Flag für 'BAM ungültig'
F09A  D0 03      BNE $F09F     hat Änderung in BAM stattgefunden ?
F09C  4C 80 A4   JMP $A58D     ja, BAM auf Diskette schreiben
F09F[1] 09 80    ORA #$80      Flag für 'BAM
F0A1  8D F9 02   STA $02F9     setzen
F0A4  60         RTS           zurück zur aufrufenden Routine
-----------------------------------------------------------------
[EF13/EF1A/F06D]
BAM-Bytes aus Zwischenspeicher in BAM kopieren
F0A5  A8         TAY           Nummer des aktuellen ZS
F0A6  B9 9D 02   LDA $029D,Y   Spurnummer des ZS holen
F0A9  F0 25      BEQ $F0D0     ist ZS belegt ?
F0AB  48         PHA           ja, Spurnummer merken
F0AC  A9 00      LDA #$00      Zwischenspeicher
F0AE  99 9D 02   STA $029D,Y   freigeben
F0B1  A5 F9      LDA $F9       Nummer des aktuellen Puffers
F0B3  0A         ASL A         verdoppeln (da Zeiger aus 2-Byte-Werten
F0B4  AA         TAX           besteht)
F0B5  68         PLA           Spurnummer wieder holen
F0B6  0A         ASL A         und vervierfachen
F0B7  0A         ASL A         (da 4 BAM-Bytes pro Spur)
F0B8  95 99      STA $99,X     Zeiger auf Spur setzen
F0BA  98         TYA           Nummer des ZS holen
F0BB  0A         ASL A         und vervierfachen
F0BC  0A         ASL A         (da vier Zwischenspeicher)
F0BD  A8         TAY           und merken
F0BE[1] B9 A1 02 LDA $02A1,Y   Byte aus BAM ZS holen
F0C1  81 99      STA ($99,X)   und in Puffer schreiben
F0C3  A9 00      LDA #$00      Wert in Zwischenspeicher
F0C5  99 A1 02   STA $02A1,Y   löschen
F0C8  F6 99      INC $99,X     Zeiger auf nächstes Byte setzen
```

```
F0CA  C8          INY            nächstes ZS Zeichen anwählen
F0CB  98          TYA            prüfen ob schon
F0CC  29 03       AND #$03       alle 4 Bytes übertragen
F0CE  D0 EE       BNE $F0BE      noch weitere Bytes kopieren ?
F0D0  60          RTS            nein, zurück zur aufrufenden Routine
--------------------------------------------------------------------
[C8F5/D042/EEB7/BF33]
Zeiger auf Position der aktuellen Spur in BAM löschen
F0D1  A5 7F       LDA $7F        aktuelles Laufwerk holen
F0D3  0A          ASL A          verdoppeln (da 2 Laufwerke möglich)
F0D4  AA          TAX            und merken
F0D5  A9 00       LDA #$00       Spurwert 0 als Flag für 'BAM-Zeiger
F0D7  9D 9D 02    STA $029D,X    inaktiv 'setzen
F0DA  E8          INX            und damit die Zeiger
F0DB  9D 9D 02    STA $029D,X    löschen
F0DE  60          RTS            zurück zur aufrufenden Routine
--------------------------------------------------------------------
[A4B0/C7BA/EF3E/F05D]
BAM von Diskette lesen
F0DF  B5 A7       LDA $A7,X      Puffernummer holen und mit Flagwert
F0E1  C9 FF       CMP #$FF       für 'Puffer frei' vergleichen
F0E3  D0 25       BNE $F10A      identisch ?
F0E5  8A          TXA            ja, Nummer des Kanals
F0E6  48          PHA            merken
F0E7  20 8E D2    JSR $D28E      Puffernummer holen
F0EA  AA          TAX            und merken
F0EB  10 05       BPL $F0F2      ist ein Puffer frei ?
F0ED  A9 70       LDA #$70       nein, Fehlermeldung
F0EF  20 C8 C1    JSR $C1C8      '70 No Channel' ausgeben
F0F2  86 F9       STX $F9        Nummer aktuellen Puffers setzen
F0F4  68          PLA            Kanalnummer wieder holen
F0F5  A8          TAY            und merken
F0F6  8A          TXA            Puffernummer holen und Flag für
F0F7  09 80       ORA #$80       'Puffer gerade nicht aktiv'
F0F9  99 A7 00    STA $00A7,Y    in Tabelle eintragen
F0FC  0A          ASL A          Puffernummer verdoppeln (da Zeiger auf
F0FD  AA          TAX            2-Byte Werte)
F0FE  AD 85 FE    LDA $FE85      Spur des Directory
F101  95 06       STA $06,X      als Spur für Job setzen
F103  A9 00       LDA #$00       Sektornummer 0
F105  95 07       STA $07,X      in für Job setzen
```

```
F107    4C 42 A5   JMP $A667      BAM von Diskette lesen
F10A[1] 29 0F      AND #$0F       Puffernummer herstellen und
F10C    85 F9      STA $F9        setzen
F10E    60         RTS            zurück zur aufrufenden Routine
---------------------------------------------------------------------
[D00E/EF3A/F022/F119]
Nummer des Kanals für BAM festlegen (in Akku)
F10F    A9 06      LDA #$06       Kanalnummer für BAM-Kanal bei Laufwerk 1
F111    A6 7F      LDX $7F        aktuelles Laufwerk holen
F113    D0 03      BNE $F118      Laufwerk 0 ?
F115    18         CLC            ja, Flag für Laufwerknummer
F116    69 07      ADC #$07       und Nummer des Kanals für BAM setzen
F118[1] 60         RTS            zurück zur aufrufenden Routine
---------------------------------------------------------------------
[A4AD/C7B7/C883/C8F8]
Nummer des Kanals für BAM festlegen (in X-Register)
F119    20 0F F1   JSR $F10F      Nummer des Kanals festlegen
F11C    AA         TAX            und in X-Register merken
F11D    60         RTS            zurück zur aufrufenden Routine
---------------------------------------------------------------------
[D1A6/DD1D/E3A9/E3BC/E44E]
Nächsten freien Block in BAM suchen
F11E    20 3E DE   JSR $DE3E      aktuelle Spur- und Sektornummer holen
F121    A9 03      LDA #$03       BAM Zeiger
F123    85 6F      STA $6F        setzen
F125    A9 01      LDA #$01       Flag für 'BAM ungültig, neue BAM
F127    0D F9 02   ORA $02F9      auf Diskette schreiben'
F12A    8D F9 02   STA $02F9      setzen
F12D[4] 4C DB A8   JMP $A8DB      nächsten freien Sektor suchen
---------------------------------------------------------------------
[A8E5/F138:A8FD,A902]
nächsten freien Sektor suchen
F130    20 11 F0   JSR $F011
F133    68         PLA            Zeiger wieder holen
F134    85 6F      STA $6F        und setzen
F136    B1 6D      LDA ($6D),Y    Zahl der noch freien Sektoren der Spur
F138[2] D0 39      BNE $F173      ist noch ein Sektor frei ?
F13A    A5 80      LDA $80        nein, aktuelle Spurnummer holen und mit
F13C    CD 85 FE   CMP $FE85      Nummer f. Directoryspur vergleichen (18)
F13F    F0 19      BEQ $F15A      identisch ?
F141    90 1C      BCC $F15F      nein, aktuelle Spur kleiner als 18 ?
```

```
F143  E6 80     INC $80     nein, Spurnummer erhöhen (Diskette wird
F145  A5 80     LDA $80     von 18 aus nach innen und außen belegt)
F147  CD AC 02  CMP $02AC   und mit maximaler Spur vergleichen
F14A  D0 E1     BNE $F12D   größte Spur erreicht ?
F14C  AE 85 FE  LDX $FE85   ja, Spuren nach außen abfahren und
F14F  CA        DEX         Directoryspur -1 als
F150  86 80     STX $80     aktuelle Spurnummer festlegen
F152  A9 00     LDA #$00    Sektorzähler
F154  85 81     STA $81     löschen
F156  C6 6F     DEC $6F     Zahl der freien Blocks
F158  D0 D3     BNE $F12D   noch ein Sektor frei ?
F15A 2 A9 72    LDA #$72    nein, Fehlermeldung
F15C  20 C8 C1  JSR $C1C8   '72 Disk Full' ausgeben
F15F 1 C6 80    DEC $80     Spur eine Spur nach außen setzen
F161  D0 CA     BNE $F12D   äußerste Spur erreicht (0) ?
F163  AE 85 FE  LDX $FE85   ja, Nummer der Directoryspur holen
F166  E8        INX         und eine Spur weiter innen als
F167  86 80     STX $80     aktuelle Spurnummer übernehmen
F169  A9 00     LDA #$00    Sektorzähler
F16B  85 81     STA $81     löschen (0)
F16D  C6 6F     DEC $6F     Zahl der freien Sektoren
F16F  D0 BC     BNE $F12D   noch ein Sektor frei ?
F171  F0 E7     BEQ $F15A   nein, Fehler 'Disk Full' ausgeben
-----------------------------------------------------------------

[F138]
Nächsten Sektor auf Spur suchen
F173  A5 81     LDA $81     Nummer des aktuellen Sektors
F175  18        CLC         dazu den optimalen Sektorversatz für
F176  65 69     ADC $69     zwei Sektoren addieren
F178  85 81     STA $81     und als aktuelle Sektornummer merken
F17A  A5 80     LDA $80     Nummer der aktuellen Spur
F17C  20 4B F2  JSR $F24B   Zahl der Sektoren, die die Spur
F17F  8D 4E 02  STA $024E   umfaßt feststellen
F182  8D 4D 02  STA $024D   und merken
F185  C5 81     CMP $81     mit der neuen Sektornummer vergleichen
F187  B0 0C     BCS $F195   ist die Nummer zu groß ?
F189  38        SEC         ja, Nummer des
F18A  A5 81     LDA $81     aktuellen Sektors holen
F18C  ED 4E 02  SBC $024E   und die maximale Sektorenanzahl abziehen
F18F  85 81     STA $81     Ergebnis als neue Sektornummer merken
F191  F0 02     BEQ $F195   ist Sektor 0 angewählt worden ?
```

```
F193  C6 81     DEC $81          nein, dann Sektorversatz korregieren
F195  20 FA F1  JSR $F1FA        nächsten freien Sektor suchen
F198  F0 03     BEQ $F19D        gefunden ?
F19A  4C 90 EF  JMP $EF90        ja, Sektor in der BAM belegen
F19D  A9 00     LDA #$00         Sektornummer 0
F19F  85 81     STA $81          setzen
F1A1  20 FA F1  JSR $F1FA        nächsten freien Sektor suchen
F1A4  D0 F4     BNE $F19A        gefunden ?
F1A6  4C F5 F1  JMP $F1F5        nein, Fehler '71 Dir Error' ausgeben
----------------------------------------------------------------------
[DCDA]
nächsten optimalen Sektor belegen
F1A9  A9 01     LDA #$01         Flag für 'BAM ungültig'
F1AB  0D F9 02  ORA $02F9        (auf Diskette zurückschreiben)
F1AE  8D F9 02  STA $02F9        setzen
F1B1  A5 86     LDA $86          Zeropageadresse wird von Routine
F1B3  48        PHA              verwendet und deshalb gerettet
F1B4  A9 01     LDA #$01         Zeiger auf Spurnummer
F1B6  85 86     STA $86          initialisieren
F1B8  AD 85 FE  LDA $FE85        Nummer der Directoryspur holen
F1BB  38        SEC              Zähler für aktuelle Spur davon abziehen
F1BC  E5 86     SBC $86          um Spurnummer ober oder unterhalb
F1BE  85 80     STA $80          Spur 18 zu erhalten und merken
F1C0  90 09     BCC $F1CB        ist Spurnummer kleiner 18 ?
F1C2  F0 07     BEQ $F1CB        nein, ist sie gleich 18 ?
F1C4  4C 05 A9  JMP $A905        nein, BAM Zeiger auf Bit des Sektors
----------------------------------------------------------------------
[A90F/F1C9:A91B]
Spur für nächsten freien Sektor festlegen
F1C7  B1 6D     LDA ($6D),Y      Zahl der freien Blocks der Spur holen
F1C9  D0 1B     BNE $F1E6        sind noch Sektoren frei ?
F1CB  AD 85 FE  LDA $FE85        nein, Nummer der Directoryspur
F1CE  18        CLC              holen und Zähler für Spur dazu addieren
F1CF  65 86     ADC $86          um Spur oberhalb der Directoryspur
F1D1  85 80     STA $80          als aktuelles Spurnummer zu erhalten
F1D3  E6 86     INC $86          Zähler für Spurnummer auf nächste Spur
F1D5  CD AC 02  CMP $02AC        mit Nummer der größten Spur vergleichen
F1D8  90 05     BCC $F1DF        maximale Spurnummer erreicht ?
F1DA  A9 67     LDA #$67         ja, Fehlermeldung
F1DC  20 45 E6  JSR $E645        '67 Illegal Track or Sector' ausgeben
F1DF  4C 1E A9  JMP $A91E        nächsten freien 1571 Sektor suchen
```

```
-----------------------------------------------------------------
[A928/F1E4:A934]
freien Sektor belegen
F1E2    B1 6D     LDA ($6D),Y    Zahl der freien Blocks der Spur holen
F1E4[1] F0 D2     BEQ $F1B8      noch ein Sektor frei ?
F1E6[1] 68        PLA            ja, Zeropageadresse $86
F1E7    85 86     STA $86        wieder einrichten
F1E9    A9 00     LDA #$00       Nummer des aktuellen Sektors
F1EB    85 81     STA $81        löschen
F1ED    20 FA F1  JSR $F1FA      und nächsten freien Sektor suchen
F1F0    F0 03     BEQ $F1F5      gefunden ?
F1F2    4C 90 EF  JMP $EF90      ja, Sektor in BAM belegen und zurück
F1F5*   A9 71     LDA #$71       Fehlermeldung
F1F7    20 45 E6  JSR $E645      '71 Dir Error' ausgeben
-----------------------------------------------------------------
[CD09/CD27/F195/F1A1/F1ED/F1FD:A8B3]
nächsten freien Sektor der Spur holen
F1FA    4C A9 A8  JMP $A8A9      nächsten freien Sektor der Spur suchen
F1FD[1] 98        TYA            Zeiger auf Position des Bitmusters der
F1FE    48        PHA            belegten Blöcke merken
F1FF    20 20 F2  JSR $F220      Zahl der freien Blocks prüfen
F202    A5 80     LDA $80        Nummer der aktuellen Spur
F204    20 4B F2  JSR $F220      Zahl der Sektoren auf dieser Spur
F207    8D 4E 02  STA $024E      holen und merken
F20A    68        PLA            Zeiger auf Bitmuster in BAM
F20B    85 6F     STA $6F        wieder holen und merken
F20D[1] A5 81     LDA $81        Nummer des aktuellen Sektors
F20F    CD 4E 02  CMP $024E      mit Gesamtzahl der Sektoren vergleichen
F212    B0 09     BCS $F21D      Sektornummer kleiner ?
F214    20 D5 EF  JSR $EFD5      ja, Bit für Sektor aus BAM holen
F217    D0 06     BNE $F21F      ist der Sektor frei ?
F219    E6 81     INC $81        nein, Zeiger auf nächsten Sektor setzen
F21B    D0 F0     BNE $F20D      immer Sprung nach $F20D
F21D[1] A9 00     LDA #$00       Flag für 'kein Sektor auf Spur frei'
F21F[1] 60        RTS            zurück zur aufrufenden Routine
-----------------------------------------------------------------
[A93E/F1FF]
Zahl der freien Blocks in BAM für jede Spur überprüfen
F220    A5 6F     LDA $6F        Zeropageadresse $6F wird als Zwischen-
F222    48        PHA            speicher verwendet und deshalb gerettet
F223    A9 00     LDA #$00       Zähler für freie Blocks
```

```
F225    85 6F       STA $6F          löschen
F227    AC 86 FE    LDY $FE86        Anzahl der BAM-Bytes pro Spur holen
F22A    88          DEY              und Zahl der Bytes für Bitmuster bilden
F22B[1] A2 07       LDX #$07         Zähler für Zahl der Bits pro Byte
F22D[2] B1 6D       LDA ($6D),Y      Byte aus BAM holen und Bit
F22F    3D E9 EF    AND $EFE9,X      auf das der Bitzähler zeigt isolieren
F232    F0 02       BEQ $F236        ist der Block belegt ?
F234    E6 6F       INC $6F          nein, Zähler der freien Blocks
F236[1] CA          DEX              erhöhen und nächstes Bit anwählen
F237    10 F4       BPL $F22D        alle Bits getestet gezählt ?
F239    88          DEY              ja, Zeiger auf nächstes BAM-Byte
F23A    D0 EF       BNE $F22B        alle BAM-Bytes der Spur geprüft ?
F23C    B1 6D       LDA ($6D),Y      ja, in BAM vermerkte Zahl der freien
F23E    C5 6F       CMP $6F          Blöcke mit errechneter Zahl vergleichen
F240    D0 04       BNE $F246        identisch ?
F242    68          PLA              ja, Zeropageadresse $6F
F243    85 6F       STA $6F          wieder einrichten
F245    60          RTS              zurück zur aufrufenden Routine
F246[1] A9 71       LDA #$71         Fehlermeldung
F248    20 45 E6    JSR $E645        '71 Dir Error' ausgeben
----------------------------------------------------------------------
[D540/D568/EED2/F17C/F204]
Anzahl der Sektoren einer Spur holen
(Im Akku muss die Nummer der Spur stehen)
F24B    20 4F A7    JSR $A74F        Zahl der Spurzonen holen
F24E[1] DD D6 FE    CMP $FED6,X      maximale Spur der Zone mit aktueller
F251    CA          DEX              Spur vergleichen und Zone wechseln
F252    B0 FA       BCS $F24E        ist die Spur größer als max. Zonenspur ?
F254    BD D1 FE    LDA $FED1,X      ja, Anzahl der Sektoren in der Spurzone
F257    60          RTS              zurück zur aufrufenden Routine
----------------------------------------------------------------------
[CB12/CDA3/E7A8]
F258    60          RTS              keine Funktion
----------------------------------------------------------------------
[BF6C/F263:A9FF]
Diskcontroller-Reset durchführen
F259    A9 6F       LDA #$6F         'Sync' und 'Write protect' als
F25B    8D 02 1C    STA $1C02        Eingangsleitungen schalten
F25E    29 F0       AND #$F0         und deren Werte im
F260    4C F8 A9    JMP $A9F8        Patch (= Korrektur)
F263[1] AD 0C 1C    LDA $1C0C        Peripheriekontrollregister setzen :
```

```
F266  29 FE     AND #$FE      CA1 'Byte Ready' auf negative Flanke
F268  09 0E     ORA #$0E      CA2 'SOE' auf Ausgang High
F26A  09 E0     ORA #$E0      CB2 (Kopf) auf Lesen schalten
F26C  8D 0C 1C  STA $1C0C     Register aktivieren
F26F  A9 41     LDA #$41      PB7 (Sync) als Ausgang und aktiviert
F271  8D 0B 1C  STA $1C0B     Eingangszwischenspeicher für Kopfdaten
F274  A9 00     LDA #$00      Zählerwert für Interrupttakt in Timer 1
F276  8D 06 1C  STA $1C06     setzen, sodaß die
F279  A9 20     LDA #$20      Diskcontrollerroutine alle 8 ms
F27B  8D 07 1C  STA $1C07     aufgerufen wird
F27E  8D 05 1C  STA $1C05     und Timer 1 starten
F281  A9 7F     LDA #$7F      Interruptflags
F283  8D 0E 1C  STA $1C0E     löschen
F286  A9 C0     LDA #$C0      Interrupt die sich durch
F288  8D 0D 1C  STA $1C0D     'Timer 1 hat Null durchlaufen'
F28B  8D 0E 1C  STA $1C0E     erlauben
F28E  A9 FF     LDA #$FF      Flags löschen :
F290  85 3E     STA $3E       Flag für aktives Laufwerk
F292  85 51     STA $51       Flag für 'Formatiervorgang' läuft
F294  A9 08     LDA #$08      Kennzeichen für Blockheader
F296  85 39     STA $39       setzen
F298  A9 07     LDA #$07      Kennzeichen für Datenblockheader
F29A  85 47     STA $47       setzen
F29C  A9 05     LDA #$05      Aufruf in $FA02 (akt. Stepperroutine)
F29E  85 62     STA $62       auf Routine
F2A0  A9 FA     LDA #$FA      in $FA05 (Steppersteuerung)
F2A2  85 63     STA $63       richten
F2A4  A9 C8     LDA #$C8      Zahl der Schritte für schnelle
F2A6  85 64     STA $64       Kopfbewegung festlegen
F2A8  A9 04     LDA #$04      Zahl der Schritte
F2AA  85 5E     STA $5E       zum Anfahren und
F2AC  A9 04     LDA #$04      Bremsen des Kopfes
F2AE  85 5F     STA $5F       festlegen
--------------------------------------------------------------------------
[9DCA/9DD5/BF06]
Einsprung in Diskcontroller-Routine
F2B0  BA        TSX           Stackpointer
F2B1  86 49     STX $49       retten
F2B3  AD 04 1C  LDA $1C04     Timer zurücksetzen
F2B6  AD 0C 1C  LDA $1C0C     CA2 (SOE = Serial Output Enable)
F2B9  09 0E     ORA #$0E      auf Ausgang High
```

```
F2BB  8D 0C 1C   STA $1C0C       setzen
F2BE  A0 05      LDY #$05        Zahl der Puffer
F2C0  B9 00 00   LDA $0000,Y     Flag für Jobcode holen
F2C3  10 2E      BPL $F2F3       liegt ein Autrag für den Puffer vor ?
F2C5  C9 D0      CMP #$D0        ja, mit 'Programm starten' vergleichen
F2C7  D0 04      BNE $F2CD       Programm im Puffer ausführen ?
F2C9  98         TYA             ja, Puffernummer holen
F2CA  4C 70 F3   JMP $F370       und Programm anspringen
F2CD  29 01      AND #$01        Laufwerksnummer aus Jobcode holen
F2CF  F0 07      BEQ $F2D8       ist der Job für Laufwerk 0 ?
F2D1  84 3F      STY $3F         nein, gerade bearbeiteter Puffer merken
F2D3  A9 0F      LDA #$0F        Fehlermeldung
F2D5  4C 69 F9   JMP $F969       '74 Drive Not Ready' ausgeben
F2D8  AA         TAX             Laufwerksnummer (0)
F2D9  85 3D      STA $3D         merken
F2DB  C5 3E      CMP $3E         und Flag für 'Laufwerk aktiv' holen
F2DD  F0 0A      BEQ $F2E9       läuft das Laufwerk schon ?
F2DF  20 7E F9   JSR $F97E       nein, Motor einschalten
F2E2  A5 3D      LDA $3D         und Flag für 'Laufwerk aktiv'
F2E4  85 3E      STA $3E         setzen
F2E6  4C 9C F9   JMP $F99C       warten, bis Motor läuft
F2E9  A5 20      LDA $20         Laufwerksstatus holen
F2EB  30 03      BMI $F2F0       ist der Motor schon auf Drehzahl ?
F2ED  0A         ASL A           ja, Flagbit für Steppermotor in Carry
F2EE  10 09      BPL $F2F9       ist der Kopf in Bewegung ?
F2F0  4C 9C F9   JMP $F99C       ja, Kopf weiter in Position bringen
F2F3  88         DEY             Puffernummer merken
F2F4  10 CA      BPL $F2C0       bereits alle Puffer überprüft ?
F2F6  4C 9C F9   JMP $F99C       ja, zur Hauptsteuerroutine
F2F9  A9 20      LDA #$20        Flag für 'Motor an'
F2FB  85 20      STA $20         als Laufwerksstatus setzen
F2FD  A0 05      LDY #$05        max. Pufferzahl
F2FF  84 3F      STY $3F         als aktuelle Puffernummer festlegen
F301  20 93 F3   JSR $F393       Zeiger auf Pufferadresse setzen
F304  30 1A      BMI $F320       liegt Jobauftrag vor ?
F306  C6 3F      DEC $3F         nein, Pufferzähler auf nächsten Puffer
F308  10 F7      BPL $F301       letzter Puffer erreicht ?
F30A  A4 41      LDY $41         ja, Nummer des letzten Jobs
F30C  20 95 F3   JSR $F395       Pufferzeiger setzen
F30F  A5 42      LDA $42         Spurdifferenz zum letzen Job holen
F311  85 4A      STA $4A         und als Zahl der auszuführenden
```

```
F313    06 4A      ASL $4A        Stepper-Halbschritte setzen
F315    A9 60      LDA #$60       Flag für Kopf in Bewegung
F317    85 20      STA $20        im Laufwerksstatus setzen
F319    B1 32      LDA ($32),Y    Nummer der Spur für Job aus Puffer
F31B    85 22      STA $22        holen und merken
F31D    4C 9C F9   JMP $F99C      Kopf auf Spur positionieren
F320[1] 29 01      AND #$01       Laufwerksnummer bilden
F322    C5 3D      CMP $3D        und mit letztem Joblaufwerk vergleichen
F324    D0 E0      BNE $F306      ist der Job für das gleiche Laufwerk ?
F326    A5 22      LDA $22        Spurnummer des letzen Jobs holen
F328    F0 12      BEQ $F33C      Spurnummer vorhanden ?
F32A    38         SEC            ja, Differenz zwischen aktueller und
F32B    F1 32      SBC ($32),Y    letzter Spur berechnen
F32D    F0 0D      BEQ $F33C      ist der Job für gleiche Spur ?
F32F    49 FF      EOR #$FF       Zahl der Stepperschritte
F331    85 42      STA $42        erzeugen und
F333    E6 42      INC $42        festlegen
F335    A5 3F      LDA $3F        Laufwerksnummer des aktuellen Jobs
F337    85 41      STA $41        übertragen
F339    4C 06 F3   JMP $F306      nächsten Puffer bearbeiten
F33C[2] A2 04      LDX #$04       Zahl der verschiedenen Spurzonen
F33E    B1 32      LDA ($32),Y    Spurnummer des Jobs holen
F340    85 40      STA $40        und merken
F342[1] DD D6 FE   CMP $FED6,X    mit größter Spur der Zone vergleichen
F345    CA         DEX            Zonenzähler auf nächste Zone setzen
F346    B0 FA      BCS $F342      liegt Spur innerhalb der Zone ?
F348    BD D1 FE   LDA $FED1,X    ja, Zahl der Sektoren der Zone holen
F34B    85 43      STA $43        und setzen
F34D    8A         TXA            Je nach Zonenzahl (0-3)
F34E    0A         ASL A          wird in den
F34F    0A         ASL A          Bits 5 und 6
F350    0A         ASL A          die Bit-Aufzeichnungsrate,
F351    0A         ASL A          mit der die Kopfelektronik auf die
F352    0A         ASL A          Diskette schreibt, bestimmt
F353    85 44      STA $44        Wert zwischenspeichern
F355    AD 00 1C   LDA $1C00      Laufwerkssteuerregister holen
F358    29 9F      AND #$9F       Bits für Bitrate löschen
F35A    05 44      ORA $44        und auf ermittelten Wert für die
F35C    8D 00 1C   STA $1C00      angewählte Zone setzen
F35F    A6 3D      LDX $3D        Nummer des Laufwerks
F361    A5 45      LDA $45        Jobcode holen
```

```
F363  C9 40     CMP #$40        mit Befehl 'Kopf auf Spur 1' vergleichen
F365  F0 15     BEQ $F37C       soll der Kopf zurückgesetzt werden ?
F367  C9 60     CMP #$60        nein, Kode für externes Jobprogramm
F369  F0 03     BEQ $F36E       Programm im Puffer einbinden ?
F36B  4C B1 F3  JMP $F3B1       nein, Kopf auf Spur setzen
-----------------------------------------------------------------
[F369]  vgl. 93A2
Programm im Puffer starten
F36E  A5 3F     LDA $3F         aktuelle Puffernummer holen
-----------------------------------------------------------------
[F2CA]
Programm starten (Pufferadresse in A)
F370  18        CLC             und daraus Highbyte
F371  69 03     ADC #$03        der absoluten Pufferadresse
F373  85 31     STA $31         berechnen und merken
F375  A9 00     LDA #$00        Lowbyte auf
F377  85 30     STA $30         Null setzen
F379  6C 30 00  JMP ($0030)     in Pufferprogramm springen
-----------------------------------------------------------------
[F365]  vgl. 93B0
Kopf auf Spur 1 zurücksetzen (BUMP = Rattern)
F37C  A9 60     LDA #$60        Flag für Kopf in Bewegung
F37E  85 20     STA $20         im Laufwerksstatus setzen
F380  AD 00 1C  LDA $1C00       Laufwerkssteuerregister
F383  29 FC     AND #$FC        holen und Stepperimpulse
F385  8D 00 1C  STA $1C00       löschen
F388  A9 A4     LDA #$A4        Zahl der Stepperschritte (92)
F38A  85 4A     STA $4A         nach außen setzen (= 46 Spuren)
F38C  A9 01     LDA #$01        Spur 1
F38E  85 22     STA $22         als Spurnummer des Jobs setzen
F390  4C 69 F9  JMP $F969       Jobschleife abschließen
-----------------------------------------------------------------
[BF0C/F301/F43A/F48F]  vgl. 93D1
Pufferzeiger für Job initialisieren
F393  A4 3F     LDY $3F         Nummer des aktuellen Puffers
-----------------------------------------------------------------
[F30C]  Pufferzeiger setzen (Puffernummer in Y)
F395  B9 00 00  LDA $0000,Y     dazugehörigen Jobcode holen
F398  48        PHA             und merken
F399  10 10     BPL $F3AB       Job vorhanden ?
F39B  29 78     AND #$78        ja, Befehlsbits für Diskcontroller
```

```
F39D   85 45      STA $45       isoliern und merken
F39F   98         TYA           Puffernummer holen
F3A0   0A         ASL A         und verdoppeln, (da 2-Byte-Werte)
F3A1   69 06      ADC #$06      dann Beginn der Spur-Sektor-Tabelle
F3A3   85 32      STA $32       einrechne und Zeiger setzen
F3A5   98         TYA           Puffernummer wieder holen
F3A6   18         CLC           und daraus pysikalische
F3A7   69 03      ADC #$03      Speicheradresse des
F3A9   85 31      STA $31       Puffers berechnen (High-Byte) und setzen
F3AB[1] A0 00     LDY #$00      Low-Byte des Zeigers
F3AD   84 30      STY $30       auf Null setzen
F3AF   68         PLA           Jobcode wieder holen
F3B0   60         RTS           zurück zur aufrufenden Routine
```

[F36B/F5E6]
Spur suchen; Dabei orientiert sich die Routine an den Informationen die in jedem Blockheader auf der Diskette stehen

```
F3B1   A2 5A      LDX #$5A      Zahl der Leseversuche (90)
F3B3   86 4B      STX $4B       festlegen
F3B5   A2 00      LDX #$00      Zähler für Zahl der Headerbytes löschen
F3B7   A9 52      LDA #$52      GCR-Kennzeichen für Blockheader
F3B9   85 24      STA $24       merken
F3BB[1] 20 56 F5  JSR $F556     auf Sync-Markierung warten
F3BE[1] 50 FE     BVC $F3BE     ist Leseelektronik bereit ?
F3C0   B8         CLV           ja, Flag wieder zurücksetzen
F3C1   AD 01 1C   LDA $1C01     Headerkennzeichen von Diskette lesen
F3C4   C5 24      CMP $24       mit Kennzeichen für Block vergleichen
F3C6   D0 3F      BNE $F407     liegt ein Blockheader vor ?
F3C8[2] 50 FE     BVC $F3C8     ja, auf nächstes Byte warten
F3CA   B8         CLV           Leseelektronik wieder aktivieren
F3CB   AD 01 1C   LDA $1C01     Byte von Diskette lesen
F3CE   95 25      STA $25,X     und in Headerpuffer speichern
F3D0   E8         INX           Zähler erhöhen
F3D1   E0 07      CPX #$07      mit Zahl der Headerbytes vergleichen
F3D3   D0 F3      BNE $F3C8     bereits ganzer Header eingelesen ?
F3D5   20 97 F4   JSR $F497     ja, Header von GCR nach Binär umwandeln
F3D8   A0 04      LDY #$04      Zeiger auf Position der Prüfsumme setzen
F3DA   A9 00      LDA #$00      Prüfsumme des Headers
F3DC[1] 59 16 00  EOR $0016,Y   berechnen
F3DF   88         DEY           Zeiger auf nächstes Byte des Headers
F3E0   10 FA      BPL $F3DC     schon alle Bytes eingerechnet ?
```

```
F3E2   C9 00      CMP #$00       ja, Wert für fehlerfreien Header
F3E4   D0 38      BNE $F41E      Prüfsummenfehler aufgetreten ?
F3E6   A6 3E      LDX $3E        nein, aktuelle Laufwerksnummer holen
F3E8   A5 18      LDA $18        Spurnummer aus gelesenem Header
F3EA   95 22      STA $22,X      als aktuelle Spur merken
F3EC   A5 45      LDA $45        Jobcode holen
F3EE   C9 30      CMP #$30       mit 'Sektor lesen' vergleichen
F3F0   F0 1E      BEQ $F410      identisch ?
F3F2   A5 3E      LDA $3E        nein, Laufwerksnummer des Jobs holen
F3F4   0A         ASL A          Zeiger auf zum Laufwerk
F3F5   A8         TAY            gehörende ID richten
F3F6   B9 12 00   LDA $0012,Y    erstes Zeichen der ID holen
F3F9   C5 16      CMP $16        und mit ID des Blockheaders vergleichen
F3FB   D0 1E      BNE $F41B      hat sich ID geändert ?
F3FD   B9 13 00   LDA $0013,Y    nein, nächstes Zeichen der ID
F400   C5 17      CMP $17        und mit ID aus Header vergleichen
F402   D0 17      BNE $F41B      identisch ?
F404   4C 23 F4   JMP $F423      ja, nächsten Job feststellen
F407[1] C6 4B     DEC $4B        Zähler für Leseversuche erniedrigen
F409   D0 B0      BNE $F3BB      90 Leseversuche ausgeführt
F40B   A9 02      LDA #$02       Fehlermeldung
F40D   20 69 F9   JSR $F969      '20 Read Error' ausgeben
F410[1] A5 16     LDA $16        ID des Blockheaders
F412   85 12      STA $12        als neue ID für
F414   A5 17      LDA $17        das aktuelle Laufwerk
F416   85 13      STA $13        übernehmen
F418[2] A9 01     LDA #$01       Nummer für 'OK'
F41A   2C         .byte $2c      nächste 2 Bytes überspringen (Bitbefehl)
F41B[2] A9 0B     LDA #$0B       Nummer für '29 Disk ID Mismatch'
F41D   2C         .byte $2c      nächste 2 Bytes überspringen (Bitbefehl)
F41E[1] A9 09     LDA #$09       Nummer für '27 Write Error'
F420   4C 69 F9   JMP $F969      Meldung zurückgeben
--------------------------------------------------------------------------
```

[F404] vgl. 94BC

nächsten otimalen Job holen

```
F423   A9 7F      LDA #$7F       Zeiger für Differenz zum nächsten Job
F425   85 4C      STA $4C        initialisieren
F427   A5 19      LDA $19        Sektornummer aus letztem Blockheader
F429   18         CLC            holen und
F42A   69 02      ADC #$02       mit maximaler
F42C   C5 43      CMP $43        Sektornummer vergleichen
```

F42E	90 02	BCC $F432	ist Nummer im erlaubten Bereich ?
F430	E5 43	SBC $43	nein, max. Sektornummer subtrahieren und
F432 [1]	85 4D	STA $4D	und neue Sektornummer merken
F434	A2 05	LDX #$05	Zahl der Puffer
F436	86 3F	STX $3F	setzen
F438	A2 FF	LDX #$FF	Wert für Pufferzeiger
F43A [1]	20 93 F3	JSR $F393	Pufferadresse setzen und Jobcode holen
F43D	10 44	BPL $F483	liegt ein Job vor ?
F43F	85 44	STA $44	ja, Jobcode merken
F441	29 01	AND #$01	und Laufwerksnummer des Jobs feststellen
F443	C5 3E	CMP $3E	mit aktuellem Laufwerk vergleichen ?
F445	D0 3C	BNE $F483	ist der Job für aktuelles Laufwerk ?
F447	A0 00	LDY #$00	ja, Pufferzeiger löschen
F449	B1 32	LDA ($32),Y	Spurnummer des Jobs
F44B	C5 40	CMP $40	mit letzter Spur vergleichen
F44D	D0 34	BNE $F483	identisch ?
F44F	A5 45	LDA $45	ja, Befehlsbits des Jobcodes holen
F451	C9 60	CMP #$60	Code für 'Programm im Puffer einbinden'
F453	F0 0C	BEQ $F461	soll Pufferprogramm ausgeführt werden ?
F455	A0 01	LDY #$01	nein, Zeiger auf Parameter für Puffer 0
F457	38	SEC	Sektornummer des Jobs
F458	B1 32	LDA ($32),Y	für Puffer 0 holen und mit
F45A	E5 4D	SBC $4D	errechnetem optimalen Sektor vergleichen
F45C	10 03	BPL $F461	ist neue Sektornummer kleiner ?
F45E	18	CLC	nein, Zahl der Sektoren bis zu diesem
F45F	65 43	ADC $43	Sektor berechnen
F461 [2]	C5 4C	CMP $4C	und mit letzter Differenz vergleichen
F463	B0 1E	BCS $F483	ist neuer Wert kleiner als letzter ?
F465	48	PHA	ja, Sektordifferenz merken
F466	A5 45	LDA $45	Befehlsbits des Jobcodes prüfen
F468	F0 14	BEQ $F47E	soll Sektor gelesen werden ?
F46A	68	PLA	nein, Differenz wieder holen
F46B	C9 09	CMP #$09	und mit 9 vergleichen
F46D	90 14	BCC $F483	ist Wert kleiner ?
F46F	C9 0C	CMP #$0C	nein, mit 13 vergleichen
F471	B0 10	BCS $F483	ist Differenz kleiner 13 ?
F473 [1]	85 4C	STA $4C	ja, neue Sektordifferenz merken
F475	A5 3F	LDA $3F	Puffernummer des Jobs
F477	AA	TAX	holen und daraus
F478	69 03	ADC #$03	die physikalische
F47A	85 31	STA $31	Speicheradresse berechnen (High-Byte)

```
F47C  D0 05     BNE $F483     immer Sprung nach $F483
F47E  68        PLA           Sektordifferenz wieder holen
F47F  C9 06     CMP #$06      und mit 6 vergleichen
F481  90 F0     BCC $F473     ist Differenz größer ?
F483  C6 3F     DEC $3F       ja, Zeiger auf nächsten Puffer richten
F485  10 B3     BPL $F43A     schon alle Puffer geprüft ?
F487  8A        TXA           ja, Puffernummer des nächsten Jobs
F488  10 03     BPL $F48D     optimalen Job gefunden ?
F48A  4C 9C F9  JMP $F99C     nein, Stepperkommandos ausführen
F48D  86 3F     STX $3F       Puffernummer merken
F48F  20 93 F3  JSR $F393     Pufferadresse berechnen
F492  A5 45     LDA $45       reinen Jobcode holen (Befehlbits)
F494  4C CA F4  JMP $F4CA     Lese- und Schreibjobs ausführen
-----------------------------------------------------------------
[F3D5]
Header von GCR-Code in Binärwerte umrechnen
F497  A5 30     LDA $30       Zeiger auf
F499  48        PHA           aktuelle
F49A  A5 31     LDA $31       Pufferadresse
F49C  48        PHA           retten
F49D  A9 24     LDA #$24      Zeiger auf
F49F  85 30     STA $30       $0024 (Beginn der Daten
F4A1  A9 00     LDA #$00      des zuletzt gelesenen Blockheaders)
F4A3  85 31     STA $31       festlegen
F4A5  A9 00     LDA #$00      Pufferzeiger für Umwandlungsroutine
F4A7  85 34     STA $34       zurücksetzen
F4A9  20 E6 F7  JSR $F7E6     5 GCR-Bytes in 4 Binärwerte umwandeln
F4A7  A5 55     LDA $55       viertes umgewandeltes Byte
F4AE  85 18     STA $18       als Spurnummer in Headerpuffer
F4B0  A5 54     LDA $54       drittes umgewandeltes Byte
F4B2  85 19     STA $19       als Sektornummer in Headerpuffer
F4B4  A5 53     LDA $53       zweites umgewandeltes Byte
F4B6  85 1A     STA $1A       als Prüfsumme in Headerpuffer
F4B8  20 E6 F7  JSR $F7E6     5 GCR-Bytes in 4 Binärwerte umwandeln
F4BB  A5 52     LDA $52       erstes umgewandeltes Byte
F4BD  85 17     STA $17       als zweites ID-Zeichen in Headerpuffer
F4BF  A5 53     LDA $53       zweites umgewandeltes Byte
F4C1  85 16     STA $16       als erstes ID-Zeichen in Headerpuffer
F4C3  68        PLA           Zeiger auf Adresse
F4C4  85 31     STA $31       des aktuellen Puffers
F4C6  68        PLA           wieder
```

```
F4C7  85 30      STA $30        herstellen
F4C9  60         RTS            zurück zur aufrufenden Routine
---------------------------------------------------------------------
[F494]  vgl. 9606
Jobcode auf 'Sektor lesen' prüfen, wenn Job identisch ausführen
F4CA  C9 00      CMP #$00       Jobcode mit Code für 'lesen' vergleichen
F4CC  F0 03      BEQ $F4D1      identisch ?
F4CE  4C 6E F5   JMP $F56E      nein, Jobcode weiter überprüfen
---------------------------------------------------------------------
Sektor lesen
F4D1  20 0A F5   JSR $F50A      Blockheader des Sektors suchen
F4D4  50 FE      BVC $F4D4      auf Byte von Diskette warten
F4D6  B8         CLV            Leseelektronik wieder bereit machen
F4D7  AD 01 1C   LDA $1C01      Byte von Kopf lesen
F4DA  91 30      STA ($30),Y    und in aktuellen Puffer schreiben
F4DC  C8         INY            Pufferzeiger auf nächstes Byte setzen
F4DD  D0 F5      BNE $F4D4      Puffer schon voll ?
F4DF  A0 BA      LDY #$BA       ja, Pufferzeiger auf Zusatzpuffer
F4E1  50 FE      BVC $F4E1      auf nächstes Byte von Diskette warten
F4E3  B8         CLV            Flag wieder bereit machen
F4E4  AD 01 1C   LDA $1C01      Byte von Lesekopf holen
F4E7  99 00 01   STA $0100,Y    und in Zusatzpuffer schreiben
F4EA  C8         INY            Pufferzeiger auf nächstes Byte setzen
F4EB  D0 F4      BNE $F4E1      Zusatzpuffer voll ?
F4ED  20 E0 F8   JSR $F8E0      ja, Sektor von GCR nach Binär wandeln
F4F0  A5 38      LDA $38        erstes Byte des Datenblock holen und
F4F2  C5 47      CMP $47        Kennzeichen für Datenblockheader
F4F4  F0 05      BEQ $F4FB      Datenblock ?
F4F6  A9 04      LDA #$04       nein, Fehlermeldung
F4F8  4C 69 F9   JMP $F969      '22 Read Error' ausgeben
F4FB  20 E9 F5   JSR $F5E9      Prüfsumme der Daten berechnen
F4FE  C5 3A      CMP $3A        mit gelesenem Wert vergleichen
F500  F0 03      BEQ $F505      identisch ?
F502  A9 05      LDA #$05       Fehlernummer für '23 Read Error'
F504  2C         .byte $2C      nächste 2 Bytes überspringen (Bitbefehl)
F505  A9 01      LDA $01        Fehlernummer für 'OK'
F507  4C 69 F9   JMP $F969      Meldung zurückgeben
---------------------------------------------------------------------
```

```
[F4D1/F6A0]  vgl. 9600
Lesekopf auf Position nach Datenblock-Sync-Markierung eines Sektors setzen
F50A  20 10 F5   JSR $F510      Blockheader des Sektors suchen
F50D  4C 56 F5   JMP $F556      auf Sync-Marke des Datenblocks warten
---------------------------------------------------------------------------
[F50A/F589/F6CA]  vgl. 970F
Sektorheader suchen
F510  A5 3D      LDA $3D        Laufwerksnummer des Jobs
F512  0A         ASL A          und zum Laufwerk
F513  AA         TAX            dazugehörige ID holen
F514  B5 12      LDA $12,X      erstes Zeichen der ID
F516  85 16      STA $16        in Headerpuffer übertragen
F518  B5 13      LDA $13,X      zweites Zeichen der ID
F51A  85 17      STA $17        in Headerpuffer übertragen
F51C  A0 00      LDY #$00       Pufferzeiger löschen
F51E  B1 32      LDA ($32),Y    Spurnummer aus aktuellem Puffer holen
F520  85 18      STA $18        und in Headerpuffer übertragen
F522  C8         INY            Pufferzeiger auf nächstes Zeichen setzen
F523  B1 32      LDA ($32),Y    Sektornummer aus aktuellem Puffer holen
F525  85 19      STA $19        und in Headerpuffer übetragen
F527  A9 00      LDA #$00       Prüfsumme
F529  45 16      EOR $16        des erstellten Sektorheaders
F52B  45 17      EOR $17        berechnen
F52D  45 18      EOR $18        und in den
F52F  45 19      EOR $19        Headerpuffer
F531  85 1A      STA $1A        schreiben
F533  20 34 F9   JSR $F934      Sektorheader in GCR-Bytes umwandeln
F536  A2 5A      LDX #$5A       Zahl der Leseversuche festlegen (90)
F538[1] 20 56 F5 JSR $F556      auf nächste Sync-Markierung warten
F53B  A0 00      LDY #$00       Pufferzeiger löschen
F53D[2] 50 FE    BVC $F53D      auf Byte von Diskette warten
F53F  B8         CLV            'Byte Ready' Flag wieder bereit machen
F540  AD 01 1C   LDA $1C01      Byte von Lesekopf holen
F543  D9 24 00   CMP $0024,Y    und mit hergestelltem Header vergleichen
F546  D0 06      BNE $F54E      Werte identisch ?
F548  C8         INY            ja, Pufferzeiger auf nächstes Zeichen
F549  C0 08      CPY #$08       mit Zahl der Headerbytes vergleichen
F54B  D0 F0      BNE $F53D      gesamter Header überprüft ?
F54D  60         RTS            ja, zurück zur aufrufenden Routine
F54E[1] CA       DEX            Zähler der Leseversuche erniedrigen
F54F  D0 E7      BNE $F538      noch einen Leseversuch durchführen ?
```

```
F551  A9 02     LDA #$02      Fehlermeldung
F553  4C 69 F9  JMP $F969     '20 Read Error' ausgeben
--------------------------------------------------------------------
[BF1E/F3BB/F50D/F538/FB1D/FD39/FD62]  vgl. 9754
auf nächste Sync-Markierung warten
F556  A9 D0     LDA #$D0      Timer auf etwa 53 ms einstellen
F558  8D 05 18  STA $1805     und starten
F55B  A9 03     LDA #$03      Nummer für '21 Read Error'
F55D  2C 05 18  BIT $1805     Zustand des Timers holen
F560  10 F1     BPL $F553     ist der Timer abgelaufen ?
F562  2C 00 1C  BIT $1C00     Zustand des Sync-Flags holen
F565  30 F6     BMI $F55D     wurde Sync-Markierung gefunden ?
F567  AD 01 1C  LDA $1C01     ja, Kopf initialisieren
F56A  B8        CLV           Leselektronik wieder bereit machen
F56B  A0 00     LDY #$00      Prozessorflags setzen
F56D  60        RTS           zurück zur aufrufenden Routine
--------------------------------------------------------------------
[F4CE]  vgl. 976E
Sektor schreiben, wenn Jobcode $90 (Befehlbits $10)
F56E  C9 10     CMP #$10      mit Jobcode für 'Schreiben' vergleichen
F570  F0 03     BEQ $F575     identisch ?
F572  4C 91 F6  JMP $F691     nein, Jobcode weiter untersuchen
--------------------------------------------------------------------
Sektor schreiben
F575  20 E9 F5  JSR $F5E9     Prüfsumme des Puffers berechnen
F578  85 3A     STA $3A       und merken
F57A  AD 00 1C  LDA $1C00     Laufwerkssteuerregister holen
F57D  29 10     AND #$10      Bitflag für 'Write Protect' holen
F57F  D0 05     BNE $F586     ist Schreibschutz vorhanden ?
F581  A9 08     LDA #$08      ja, Fehlernummer
F583  4C 69 F9  JMP $F969     '26 Write Protect On' ausgeben
F586  20 8F F7  JSR $F78F     Puffer in GCR-Code umwandeln
F589  20 10 F5  JSR $F510     Blockheader des Sektors suchen
F58C  A2 09     LDX #$09      Zahl der Bytes des Headers
F58E  50 FE     BVC $F58E     Byte von Diskette gelesen ?
F590  B8        CLV           ja, Byte Ready wieder bereit machen
F591  CA        DEX           nächstes Byte überlesen
F592  D0 FA     BNE $F58E     gesamter Blockheader übersprungen ?
F594  A9 FF     LDA #$FF      ja, Register für Kopf auf Ausgang
F596  8D 03 1C  STA $1C03     schalten
F599  AD 0C 1C  LDA $1C0C     Laufwerkssteuetrregister holen
```

```
F59C   29 1F       AND #$1F       Kontrollerelektronik auf Schreiben
F59E   09 C0       ORA #$C0       stellen
F5A0   8D 0C 1C    STA $1C0C      und in Register setzen
F5A3   A9 FF       LDA #$FF       Wert für Sync-Markierung
F5A5   A2 05       LDX #$05       Zahl der Sync-Bytes für Markierung
F5A7   8D 01 1C    STA $1C01      Byte an Kopf übertragen
F5AA   B8          CLV            Flag für Byte Ready bereit machen
F5AB[2] 50 FE      BVC $F5AB      warten bis Byte geschrieben ist
F5AD   B8          CLV            Flag für Byte Ready bereit machen
F5AE   CA          DEX            Zähler für Zahl der Sync-Bytes
F5AF   D0 FA       BNE $F5AB      alle Sync-Bytes auf Diskette ?
F5B1   A0 BB       LDY #$BB       ja, Pufferzeiger auf Zusatzpuffer
F5B3[1] B9 00 01   LDA $0100,Y    Byte aus Puffer holen
F5B6[1] 50 FE      BVC $F5B6      warten bis Schreiblektronik bereit ist
F5B8   B8          CLV            Flag wieder zurücksetzen
F5B9   8D 01 1C    STA $1C01      Byte auf Diskette schreiben
F5BC   C8          INY            Zeiger auf nächstes Zeichen in Puffer
F5BD   D0 F4       BNE $F5B3      gesamter Puffer geschrieben ?
F5BF[1] B1 30      LDA ($30),Y    ja, Byte aus Datenpuffer holen
F5C1[1] 50 FE      BVC $F5C1      warten bis Diskette bereit ist
F5C3   B8          CLV            Flag wieder setzen
F5C4   8D 01 1C    STA $1C01      und Byte auf Diskette schreiben
F5C7   C8          INY            Zeiger auf nächstes Byte im Puffer
F5C8   D0 F5       BNE $F5BF      bereits ganzer Puffer geschrieben ?
F5CA[1] 50 FE      BVC $F5CA      ja, warten bis letztes Byte vollständig
F5CC   AD 0C 1C    LDA $1C0C      geschrieben ist und dann
F5CF   09 E0       ORA #$E0       Kontrollerelektronik wieder
F5D1   8D 0C 1C    STA $1C0C      auf Lesen umschalten
F5D4   A9 00       LDA #$00       Register des Lesekopf auf Eingang
F5D6   8D 03 1C    STA $1C03      schalten
F5D9   20 F2 F5    JSR $F5F2      Puffer von GCR in Binär zurückwandeln
F5DC   A4 3F       LDY $3F        aktuelle Puffernummer
F5DE   B9 00 00    LDA $0000,Y    dazugehörigen Jobcode holen
F5E1   49 30       EOR #$30       und daraus Jobcode für
F5E3   99 00 00    STA $0000,Y    'Verify' herstellen
F5E6   4C B1 F3    JMP $F3B1      Überprüfung durchführen
------------------------------------------------------------------------

[96FD/9775/989E/9C1B/BF2A/F4FB/F575/F698/FCA2]
Prüfsumme des Puffers berechnen
F5E9   A9 00       LDA #$00       Wert für Prüfsumme und
F5EB   A8          TAY            Zeiger auf Pufferposition löschen
```

```
F5EC[1] 51 30     EOR ($30),Y     Byte aus Puffer Prüfsumme einrechnen
F5EE  C8          INY             Zeiger auf nächstes Byte setzen
F5EF  D0 FB       BNE $F5EC       bereits ganzer Puffer eingerechnet ?
F5F1  60          RTS             ja, zurück zur aufrufenden Routine
--------------------------------------------------------------------
[F5D9/F972]  vgl. 97F9
Datenpuffer und Zusatzpuffer von GCR-Code in Binärwerte umwandeln
F5F2  A9 00       LDA #$00        Low-Byte des Zeigers für den
F5F4  85 2E       STA $2E         aktuellen Datenpuffer und den
F5F6  85 30       STA $30         Zusatzpuffer initialisieren
F5F8  85 4F       STA $4F         momentanen Wert des Zeigers auf den
F5FA  A5 31       LDA $31         aktuellen Datenpuffer
F5FC  85 4E       STA $4E         in $4E/$4F retten
F5FE  A9 01       LDA #$01        Pufferzeiger
F600  85 31       STA $31         auf $1BB setzen
F602  85 2F       STA $2F         High-Byte auf Zusatzpuffer
F604  A9 BB       LDA #$BB        Pufferzeiger für Umwandlungsroutine auf
F606  85 34       STA $34         Beginn des Zusatzpuffers richten
F608  85 36       STA $36         Zeiger auf akt. Binärbyteposition setzen
F60A  20 E6 F7    JSR $F7E6       5 GCR-Bytes in 4 Binärwerte umwandeln
F60D  A5 52       LDA $52         erstes umgewandeltes Byte holen und als
F60F  85 38       STA $38         Kennzeichen für Datenblockheader merken
F611  A4 36       LDY $36         Pufferzeiger wieder holen
F613  A5 53       LDA $53         zweites umgwandeltes Byte holen
F615  91 2E       STA ($2E),Y     und in Zwischenpuffer schreiben
F617  C8          INY             Pufferzeiger auf nächstes Byte setzen
F618  A5 54       LDA $54         drittes umgewandeltes Byte holen
F61A  91 2E       STA ($2E),Y     und in Zwischenpuffer schreiben
F61C  C8          INY             Zeiger auf nächstes Byte
F61D  A5 55       LDA $55         letztes umgewandeltes Byte holen
F61F  91 2E       STA ($2E),Y     und in Zwischenpuffer speichern
F621  C8          INY             Zeiger auf nächstes Position im Puffer
F622  84 36       STY $36         und merken
F624[1] 20 E6 F7  JSR $F7E6       die nächsten 5 GCR-Bytes umwandeln
F627  A4 36       LDY $36         Pufferzeiger wieder holen
F629  A5 52       LDA $52         erstes umgewandeltes Byte holen
F62B  91 2E       STA ($2E),Y     und in Zwischenpuffer schreiben
F62D  C8          INY             Zeiger auf nächstes Byte setzen
F62E  A5 53       LDA $53         zweites umgewandeltes Byte holen
F630  91 2E       STA ($2E),Y     und in Zwischenpuffer schreiben
F632  C8          INY             Zeiger auf nächstes Byte setzen
```

```
F633  F0 0E     BEQ $F643      alle Bytes des Zwischenpuffers geholt ?
F635  A5 54     LDA $54        nein, drittes umgewandeltes Byte holen
F637  91 2E     STA ($2E),Y    und in Zwischenpuffer schreiben
F639  C8        INY            Pufferzeiger auf nächste Byteposition
F63A  A5 55     LDA $55        viertes umgewandeltes Byte holen
F63C  91 2E     STA ($2E),Y    und in Zwischenpuffer schreiben
F63E  C8        INY            Zeiger auf nächstes Byte in Puffer
F63F  84 36     STY $36        und merken
F641  D0 E1     BNE $F624      letztes Byte aus Zwischenpuffer ?
F643  A5 54     LDA $54        ja, drittes umgewandeltes Byte holen
F645  91 30     STA ($30),Y    und in Datenpuffer schreiben
F647  C8        INY            Pufferzeiger auf nächstes Byte setzen
F648  A5 55     LDA $55        letztes umgewandeltes Byte holen
F64A  91 30     STA ($30),Y    und in Datenpuffer schreiben
F64C  C8        INY            Pufferzeiger auf nächstes Zeichen
F64D  84 36     STY $36        setzen und merken
F64F  20 E6 F7  JSR $F7E6      die nächsten 5 GCR-Bytes in Binär
F652  A4 36     LDY $36        Pufferzeiger wieder holen
F654  A5 52     LDA $52        erstes umgewandeltes Byte holen
F656  91 30     STA ($30),Y    und in Datenpuffer schreiben
F658  C8        INY            Pufferzeiger auf nächstes Byte setzen
F659  A5 53     LDA $53        zweites umgewandeltes Byte holen
F65B  91 30     STA ($30),Y    in Datenpuffer schreiben
F65D  C8        INY            Pufferzeiger korregieren
F65E  A5 54     LDA $54        drittes umgewandeltes Byte holen
F660  91 30     STA ($30),Y    in Datenpuffer schreiben
F662  C8        INY            Zeiger auf nächstes Byte im Puffer
F663  A5 55     LDA $55        letztes umgewandeltes Byte holen
F665  91 30     STA ($30),Y    in Datenpuffer schreiben
F667  C8        INY            Pufferzeiger auf nächstes Byte setzen
F668  84 36     STY $36        und merken
F66A  C0 BB     CPY #$BB       Pufferzeiger mit Endwert vergleichen
F66C  90 E1     BCC $F64F      alle Bytes in Binär umgewandelt ?
F66E  A9 45     LDA #$45       ja, Zeiger auf
F670  85 2E     STA $2E        Zieladresse der
F672  A5 31     LDA $31        nachfolgenden Verschiebeoperationen
F674  85 2F     STA $2F        setzen
F676  A0 BA     LDY #$BA       Pufferzeiger auf Beginn des Datenpuffers
F678  B1 30     LDA ($30),Y    Byte aus unterm Teil des Puffers holen
F67A  91 2E     STA ($2E),Y    und in den oberen Teil verschieben
F67C  88        DEY            Zeiger auf nächstes Zeichen
```

```
F67D  D0 F9     BNE $F678      gesamter unterer Teil kopiert ?
F67F  B1 30     LDA ($30),Y    letztes Byte in
F681  91 2E     STA ($2E),Y    oberen Teil kopieren
F683  A2 BB     LDX #$BB       Pufferzeiger für Zusatzpuffer setzen
F685  BD 00 01  LDA $0100,X    Byte aus Zusatzpuffer holen und in
F688  91 30     STA ($30),Y    unteren, freigewordenen Datenpuffer
F68A  C8        INY            Zusatzpufferzeiger erhöhen und Puffer-
F68B  E8        INX            zeiger des  Datenpuffers erhöhen
F68C  D0 F7     BNE $F685      gesamter Zusatzpuffer im Datenpuffer ?
F68E  86 50     STX $50        ja, Flag für 'Puffer im GCR-Code'
F690  60        RTS            löschen; zurück zur aufrufenden Routine
---------------------------------------------------------------------

[F572]  vgl. 9898
Sektor auf Diskette mit Pufferinhalt vergleichen, wenn Jobcode $A0
F691  C9 20     CMP #$20       Jobcode m. Code für 'Verify' vergleichen
F693  F0 03     BEQ $F698      identisch ?
F695  4C CA F6  JMP $F6CA      nein, Jobcode weiter dekodieren
---------------------------------------------------------------------

Sektor verify
F698  20 E9 F5  JSR $F5E9      Prüfsumme des Datenpuffers berechnen
F69B  85 3A     STA $3A        und merken
F69D  20 8F F7  JSR $F78F      Puffer in GCR-Code umwandeln
F6A0  20 0A F5  JSR $F50A      Kopf auf Sektoranfang auf Disk setzen
F6A3  A0 BB     LDY #$BB       Pufferzeiger auf Beginn des Zusatzpuffer
F6A5  B9 00 01  LDA $0100,Y    Byte aus Zusatzpuffer holen
F6A8  50 FE     BVC $F6A8      warten bis Byte von Diskette fertig
F6AA  B8        CLV            Flag wieder bereit machen
F6AB  4D 01 1C  EOR $1C01      Byte vom Kopf holen und vergleichen
F6AE  D0 15     BNE $F6C5      Byte von Puffer und Diskette gleich ?
F6B0  C8        INY            ja, Zeiger auf nächstes Pufferbyte
F6B1  D0 F2     BNE $F6A5      bereits ganzer Zusatzpuffer verglichen ?
F6B3  B1 30     LDA ($30),Y    ja, Byte aus Datenpuffer holen
F6B5  50 FE     BVC $F6B5      warten bis Byte von Disk gelesen
F6B7  B8        CLV            und Kopf wieder bereit machen
F6B8  4D 01 1C  EOR $1C01      Byte vom Kopf holen und vergleichen
F6BB  D0 08     BNE $F6C5      Byte von Diskette und Puffer gleich ?
F6BD  C8        INY            ja, Pufferzeiger auf nächstes Zeichen
F6BE  C0 FD     CPY #$FD       mit Endwert des Puffers vergleichen
F6C0  D0 F1     BNE $F6B3      alle Bytes verglichen ?
F6C2  4C 18 F4  JMP $F418      Verify erfolgreich
F6C5  A9 07     LDA #$07       Fehlermeldung
```

```
F6C7  4C 69 F9   JMP $F969       '25 Write Error' ausgeben
-----------------------------------------------------------------------
[F695]  vgl. 98CE
Sekorheader suchen (Jobcode $B0)
F6CA  20 10 F5   JSR $F510       Sektorheader suchen
F6CD  4C 18 F4   JMP $F418       Rückmeldung bereitstellen
-----------------------------------------------------------------------
[F7E3/FE64/F7BC/F950/F961/FE5E]
Wandelt 4 Binärbytes in 5 GCR-Bytes um. Dabei wird $52-$55 als Puffer
für die Binärwerte verwendet
F6D0  A9 00      LDA #$00        Zwischenspeicher
F6D2  85 57      STA $57         für GCR-Bytes
F6D4  85 5A      STA $5A         löschen
F6D6  A4 34      LDY $34         Zeiger auf aktuelles GCR-Byte
F6D8  A5 52      LDA $52         erstes umzuwandelndes
F6DA  29 F0      AND #$F0        Zeichen aus Binärpuffer
F6DC  4A         LSR A           holen,
F6DD  4A         LSR A           hoherwertiger Teil (Bit 4-7) des
F6DE  4A         LSR A           Bytes isolieren und in
F6DF  4A         LSR A           niederwertigeren Teil (Bit 0-3) kopieren
F6E0  AA         TAX             dann dem Halbbytes entsprechenden
F6E1  BD 7F F7   LDA $F77F,X     5-Bit-GCR-Code holen
F6E4  0A         ASL A           und die 5 Bits wieder in den höheren
F6E5  0A         ASL A           Teil des Bytes (Bit 3-7)
F6E6  0A         ASL A           kopieren und
F6E7  85 56      STA $56         merken
F6E9  A5 52      LDA $52         erstes umzuwandeltes Byte nochmal holen
F6EB  29 0F      AND #$0F        und nun niederen Teil isolieren
F6ED  AA         TAX             dann den zum Halbbyte passenden
F6EE  BD 7F F7   LDA $F77F,X     5-Bit-GCR-Code holen
F6F1  6A         ROR A           niederste zwei Bits, da im ersten Byte
F6F2  66 57      ROR $57         kein Platz mehr ist (aus 8 Bits werden
F6F4  6A         ROR A           10 Bits) in zweitem GCR-Byte
F6F5  66 57      ROR $57         unterbringen
F6F7  29 07      AND #$07        die drei verbleibenden Bits in das erste
F6F9  05 56      ORA $56         GCR-Byte einblenden
F6FB  91 30      STA ($30),Y     und GCR-Byte in Puffer schreiben
F6FD  C8         INY             Pufferzeiger auf nächstes Zeichen
F6FE  A5 53      LDA $53         zweites umzuwandelndes Byte holen
F700  29 F0      AND #$F0        ersten umzuwandelnden Teil holen
F702  4A         LSR A           und in niederwertige
```

```
F703  4A        LSR A        Bytehälfte schieben
F704  4A        LSR A        um Byte als Zeiger auf äquvalentes
F705  4A        LSR A        Binärbyte zu verwenden
F706  AA        TAX          dem Halbbyte entsprechenden
F707  BD 7F F7  LDA $F77F,X  5-Bit-GCR-Code holen
F70A  0A        ASL A        und in zweites GCR-Byte
F70B  05 57     ORA $57      in die Bitpositionen 1-5
F70D  85 57     STA $57      setzen
F70F  A5 53     LDA $53      drittes umzuwandelndes Byte holen
F711  29 0F     AND #$0F     und niederwertigen Teil isolieren
F713  AA        TAX          dann dazugehöriges
F714  BD 7F F7  LDA $F77F,X  5-Bit-GCR-Byte holen
F717  2A        ROL A        GCR-Byte in
F718  2A        ROL A        die Bitposition 4-7
F719  2A        ROL A        des dritten GCR-Bytes
F71A  2A        ROL A        setzen
F71B  85 58     STA $58      und merken
F71D  2A        ROL A        letztes GCR-Bit in
F71E  29 01     AND #$01     nächstes GCR-Byte
F720  05 57     ORA $57      übertragen
F722  91 30     STA ($30),Y  GCR-Byte in Puffer schreiben
F724  C8        INY          Pufferzeiger auf nächstes Byte setzen
F725  A5 54     LDA $54      drittes umzuwandelndes Binärbyte holen
F727  29 F0     AND #$F0     und höherwertigen Teil (Bit 4-7)
F729  4A        LSR A        isolieren
F72A  4A        LSR A        Bytehälfte in niederwertigen Teil des
F72B  4A        LSR A        Bytes verschieben und so Zeiger auf
F72C  4A        LSR A        äuivalentes Binärbyte herstellen
F72D  AA        TAX          und dem Halbbyte entsprechenden
F72E  BD 7F F7  LDA $F77F,X  5-Bit-GCR-Code holen
F731  18        CLC          Byte eins nach recht schieben und
F732  6A        ROR A        Nullbit einfügen
F733  05 58     ORA $58      GCR-Wert mit vorherigem verknüpfen
F735  91 30     STA ($30),Y  GCR-Byte in Puffer schreiben
F737  C8        INY          und Pufferzeiger erhöhen
F738  6A        ROR A        vorher herausgeschobenes Bit0 wieder
F739  29 80     AND #$80     holen und in nächstes GCR-Byte
F73B  85 59     STA $59      übernehmen
F73D  A5 54     LDA $54      niederwertigen Teil (Bit 0-3)
F73F  29 0F     AND #$0F     des dritten umzuwandelnden Bytes
F741  AA        TAX          herstellen und dazu passenden
```

```
F742  BD 7F F7   LDA $F77F,X     5-Bit-GCR-Code feststellen
F745  0A         ASL A           GCR-Wert in Position
F746  0A         ASL A           2 bis 6 setzen
F747  29 7C      AND #$7C        und als zweiten Teil des vierten
F749  05 59      ORA $59         GCR-Bytes merken
F74B  85 59      STA $59         GCR-Byte merken
F74D  A5 55      LDA $55         viertes umzuwandelndes Binärbyte holen
F74F  29 F0      AND #$F0        und höherwertigen Teil (Bit 4-7)
F751  4A         LSR A           isolieren
F752  4A         LSR A           Halbbyte in niederwertigere Bytehälfte
F753  4A         LSR A           verschieben da Byte als Zeiger
F754  4A         LSR A           auf passenden GCR-Wert diehnt
F755  AA         TAX             dann den dem Binärbyte entsprechenden
F756  BD 7F F7   LDA $F77F,X     5-Bit-GCR-Code holen
F759  6A         ROR A           erste drei Bits des
F75A  66 5A      ROR $5A         GCR-Werts (Position 0-2)
F75C  6A         ROR A           in die Position 5-7
F75D  66 5A      ROR $5A         des letzten GCR-Werts
F75F  6A         ROR A           transferieren
F760  66 5A      ROR $5A         (über Carry)
F762  29 03      AND #$03        restliche zwei Bits
F764  05 59      ORA $59         in vorhergehenden GCR-Wert einblenden
F766  91 30      STA ($30),Y     und in Puffer schreiben
F768  C8         INY             Pufferzeiger auf nächstes Byte setzen
F769  D0 04      BNE $F76F       Ende des Puffers erreicht ?
F76B  A5 2F      LDA $2F         ja, Zeiger auf Datenpuffer wieder
F76D  85 31      STA $31         setzen
F76F  A5 55      LDA $55         letztes Halbbyte aus
F771  29 0F      AND #$0F        letzem Binärbyte holen
F773  AA         TAX             und merken
F774  BD 7F F7   LDA $F77F,X     dazu passenden GCR-Wert ermitteln
F777  05 5A      ORA $5A         und in letztes GCR-Byte einblenden
F779  91 30      STA ($30),Y     Byte in Puffer schreiben
F77B  C8         INY             Pufferzeiger auf nächstes Byte setzen
F77C  84 34      STY $34         und merken
F77E  60         RTS             zurück zur aufrufenden Routine
------------------------------------------------------------------------
[F6E1/F6EE/F707/F714/F72E/F742/F756/F774]
F77F  0A 0B 12 13 0E 0F 16 17    Tabelle der den 16 Halbbytes
F787  09 19 1A 1B 0D 1D 1E 15    entsprechenden 5-Bit-GCR-Bytes
------------------------------------------------------------------------
```

[9706/9BA3/9C20/F586/F69D/FCA7]
Pufferinhalt von Binär- in GCR-Bytes umwandeln

```
F78F  A9 00     LDA #$00     Low-Bytes der Zeiger auf Null setzen :
F791  85 30     STA $30      Zeiger auf aktuellen GCR-Puffer
F793  85 2E     STA $2E      Zeiger auf aktuellen Binärpuffer
F795  85 36     STA $36      Zeiger in aktuelle Pufferposition
F797  A9 BB     LDA #$BB     Low-Byte für Zeiger auf
F799  85 34     STA $34      Zusatzpuffer setzen
F79B  85 50     STA $50      Flag für 'Puffer in GCR-Code' setzen
F79D  A5 31     LDA $31      Zeiger auf aktuellen Datenpuffer
F79F  85 2F     STA $2F      setzen
F7A1  A9 01     LDA #$01     Zeiger auf Zusatzpuffer (High-Byte)
F7A3  85 31     STA $31      einrichten
F7A5  A5 47     LDA $47      Kennzeichen für Datenblock
F7A7  85 52     STA $52      als erstes umzurechnendes Zeichen setzen
F7A9  A4 36     LDY $36      Pufferzeiger holen
F7AB  B1 2E     LDA ($2E),Y  Datenbyte aus Puffer holen und als
F7AD  85 53     STA $53      erstes umzuwandelndes Zeichen merken
F7AF  C8        INY          Pufferzeiger erhöhen
F7B0  B1 2E     LDA ($2E),Y  nächstes Datenbyte holen und als
F7B2  85 54     STA $54      zweites umzuwandeldes Byte merken
F7B4  C8        INY          Pufferzeiger auf nächstes Zeichen setzen
F7B5  B1 2E     LDA ($2E),Y  und Byte aus Datenpuffer holen und als
F7B7  85 55     STA $55      drittes umzuwandelndes Byte speichern
F7B9  C8        INY          Pufferzeiger auf nächstes Byte setzen
F7BA  84 36     STY $36      und merken
F7BC  20 D0 F6  JSR $F6D0    4 Binärbytes in 5 GCR-Bytes umwandeln
F7BF  A4 36     LDY $36      Pufferzeiger wieder holen
F7C1  B1 2E     LDA ($2E),Y  nächstes umzuwandelndes Byte holen
F7C3  85 52     STA $52      und im Zwischenspeicher merken
F7C5  C8        INY          Pufferzeiger auf nächstes Zeichen setzen
F7C6  F0 11     BEQ $F7D9    Ende des Zwischenpuffers erreicht ?
F7C8  B1 2E     LDA ($2E),Y  zweites umzuwandelndes Datenbyte holen
F7CA  85 53     STA $53      und merken
F7CC  C8        INY          Pufferzeiger erhöhen
F7CD  B1 2E     LDA ($2E),Y  drittes umzuewandelndes Byte holen
F7CF  85 54     STA $54      und in GCR-Puffer speichern
F7D1  C8        INY          Pufferzeiger auf nächstes Byte setzen
F7D2  B1 2E     LDA ($2E),Y  viertes umzuwandelnndes Byte holen
F7D4  85 55     STA $55      und merken
F7D6  C8        INY          Pufferzeiger auf nächstes Zeichen setzen
```

```
F7D7  D0 E1     BNE $F7BA      ganzer Puffer umgewandelt ?
F7D9  A5 3A     LDA $3A        Prüfsumme des Datenblocks
F7DB  85 53     STA $53        merken
F7DD  A9 00     LDA #$00       und restlichen GCR-Arbeitspuffer
F7DF  85 54     STA $54        mit Füllwerten
F7E1  85 55     STA $55        belegen
F7E3  4C D0 F6  JMP $F6D0      4 Binärbytes in 5 GCR-Werte umrechnen
------------------------------------------------------------------------
[BF2D/F4A9/F4B8/F60A/F624/F64F/F8F4/F90E]  vgl. 98D9
5 GCR-Bytes in 4 Binärbytes umwandeln
F7E6  A4 34     LDY $34        Pufferzeiger wieder holen
F7E8  B1 30     LDA ($30),Y    erstes Byte aus Puffer holen
F7EA  29 F8     AND #$F8       und 5-Bit-GCR-Wert
F7EC  4A        LSR A          isolieren
F7ED  4A        LSR A          dann daraus 8-Bit-Wert herstellen,
F7EE  4A        LSR A          wobei die 5 GCR-Bits die
F7EF  85 56     STA $56        Positionen 0-4 belegen und Wert merken
F7F1  B1 30     LDA ($30),Y    zweites Byte aus Puffer holen
F7F3  29 07     AND #$07       und 3 Bits des nächsten GCR-Bytes
F7F5  0A        ASL A          isolieren und in die Bitposition 0-4
F7F6  0A        ASL A          verschieben
F7F7  85 57     STA $57        entstandenes reines GCR-Byte merken
F7F9  C8        INY            Pufferzeiger auf nächstes Zeichen setzen
F7FA  D0 06     BNE $F802      Ende des Puffers erreicht ?
F7FC  A5 4E     LDA $4E        Zeiger auf aktuellen
F7FE  85 31     STA $31        Datenpuffer herstellen
F800  A4 4F     LDY $4F        Zeiger auf Pufferposition
F802  B1 30     LDA ($30),Y    zweites GCR-Byte holen
F804  29 C0     AND #$C0       und restliche 2 Bits des GCR-Werts
F806  2A        ROL A          holen und in richtige
F807  2A        ROL A          Position (Bit0-1)
F808  2A        ROL A          verschieben
F809  05 57     ORA $57        ersten Teil einblenden
F80B  85 57     STA $57        GCR-Wert merken
F80D  B1 30     LDA ($30),Y    GCR-Byte aus Puffer holen
F80F  29 3E     AND #$3E       nächsten GCR-Wert herstellen
F811  4A        LSR A          und in Ausgangsposition setzen
F812  85 58     STA $58        Wert merken
F814  B1 30     LDA ($30),Y    nächstes GCR-Byte holen
F816  29 01     AND #$01       und ersten Teil des nächsten GCR-Werts
F818  0A        ASL A          holen
```

```
F819  0A       ASL A          Bit in Position 4
F81A  0A       ASL A          des Bytes
F81B  0A       ASL A          verschieben
F81C  85 59    STA $59        und Wert merken
F81E  C8       INY            Zeiger auf nächstes Byte setzen
F81F  B1 30    LDA ($30),Y    und GCR-Byte aus Puffer holen
F821  29 F0    AND #$F0       zweiten Teil des GCR-Werts
F823  4A       LSR A          holen und
F824  4A       LSR A          in die untere Hälfte
F825  4A       LSR A          des Bytes (Position 0-3)
F826  4A       LSR A          verschieben
F827  05 59    ORA $59        vorheriges Bit einblenden
F829  85 59    STA $59        und GCR-Wert merken
F82B  B1 30    LDA ($30),Y    GCR-Byte nochmal aus Puffer holen
F82D  29 0F    AND #$0F       und jetzt die ersten 4 Bits
F82F  0A       ASL A          des nächsten GCR-Werts holen
F830  85 5A    STA $5A        und merken
F832  C8       INY            Pufferzeiger auf nächstes Byte
F833  B1 30    LDA ($30),Y    GCR-Byte aus Puffer holen
F835  29 80    AND #$80       und letztes Bit des
F837  18       CLC            vorhergehenden
F838  2A       ROL A          GCR-Wertes holen
F839  2A       ROL A          Bit in Position 0 des Bytes
F83A  29 01    AND #$01       verschieben und
F83C  05 5A    ORA $5A        vorherige 4 Bits einblenden
F83E  85 5A    STA $5A        GCR-Wert merken
F840  B1 30    LDA ($30),Y    GCR-Byte nochmal aus Puffer holen
F842  29 7C    AND #$7C       GCR-Wert isolieren
F844  4A       LSR A          und in Position 0-4
F845  4A       LSR A          des Bytes verschieben
F846  85 5B    STA $5B        Wert merken
F848  B1 30    LDA ($30),Y    GCR-Byte nochmal holen
F84A  29 03    AND #$03       und 2 Bits des
F84C  0A       ASL A          nächsten GCR-Werts holen
F84D  0A       ASL A          Bits in Position 3 und 4
F84E  0A       ASL A          verschieben
F84F  85 5C    STA $5C        Wert merken
F851  C8       INY            Pufferzeiger auf nächstes Byte
F852  D0 06    BNE $F85A      Ende des Puffers erreicht ?
F854  A5 4E    LDA $4E        Pufferzeiger auf aktuellen
F856  85 31    STA $31        Datenpuffer richten
```

```
F858  A4 4F       LDY $4F         Positionszeiger wieder holen
F85A  B1 30       LDA ($30),Y     GCR-Byte aus Puffer lesen
F85C  29 E0       AND #$E0        und restliche 3 Bits des vorhergehenden
F85E  2A          ROL A           GCR-Werts isolieren
F85F  2A          ROL A           Bits in Position 0-2
F860  2A          ROL A           verschieben
F861  2A          ROL A           (durch Carry)
F862  05 5C       ORA $5C         vorherige 2 Bits einblenden
F864  85 5C       STA $5C         reinen GCR-Wert merken
F866  B1 30       LDA ($30),Y     Byte aus GCR-Puffer holen
F868  29 1F       AND #$1F        letzten GCR-Wert isolieren
F86A  85 5D       STA $5D         und merken
F86C  C8          INY             Pufferzeiger auf nächstes Byte
F86D  84 34       STY $34         und merken
F86F  A6 56       LDX $56         erstes 5-Bit-GCR-Byte laden
F871  BD A0 F8    LDA $F8A0,X     und äquivalenten höherwertigen Teil
F874  A6 57       LDX $57         mit dem niederwertigen Teil, der durch
F876  1D C0 F8    ORA $F8C0,X     das zweite GCR-Byte bestimmt wird,
F879  85 52       STA $52         verknüpfen und als Binärbyte merken
F87B  A6 58       LDX $58         drittes 5-Bit-GCR-Byte laden
F87D  BD A0 F8    LDA $F8A0,X     und äquivalenten höherwertigen Teil
F880  A6 59       LDX $59         mit dem niederwertigen Teil, der durch
F882  1D C0 F8    ORA $F8C0,X     das vierte GCR-Byte bestimmt wird,
F885  85 53       STA $53         verknüpfen und als Binärbyte merken
F887  A6 5A       LDX $5A         fünftes 5-Bit-GCR-Byte laden
F889  BD A0 F8    LDA $F8A0,X     und äquivalenten höherwertigen Teil
F88C  A6 5B       LDX $5B         mit dem niederwertigen Teil, der durch
F88E  1D C0 F8    ORA $F8C0,X     das sechste GCR-Byte bestimmt wird,
F891  85 54       STA $54         verknüpfen und als Binärbyte merken
F893  A6 5C       LDX $5C         siebtes 5-Bit-GCR-Byte laden
F895  BD A0 F8    LDA $F8A0,X     und äquivalenten höherwertigen Teil
F898  A6 5D       LDX $5D         mit dem niederwertigen Teil, der durch
F89A  1D C0 F8    ORA $F8C0,X     das achte GCR-Byte bestimmt wird,
F89D  85 55       STA $55         verknüpfen und als Binärbyte merken
F89F  60          RTS             zurück zur aufrufenden Routine
----------------------------------------------------------------------
F8A0  FF FF FF FF FF FF FF FF     Tabelle der höherwertige Teile
F8A8  FF 80 00 10 FF C0 40 50     der zu den GCR-Werten äquivalenten
F8B0  FF FF 20 30 FF F0 60 70     Binärbytes; $FF bedeutet, daß dieser
F8B8  FF 90 A0 B0 FF D0 E0 FF     GCR-Wert nicht definiert ist
----------------------------------------------------------------------
```

```
F8C0  FF FF FF FF FF FF FF FF   Tabelle der niederwertigen Teile
F8C8  FF 08 00 01 FF 0C 04 05   der zu den GCR-Werten äquivalenten
F8D0  FF FF 02 03 FF 0F 06 07   Binärbytes; $FF bedeutet, daß dieser
F8D8  FF 09 0A 0B FF 0D 0E FF   GCR-Wert nicht definiert ist
-----------------------------------------------------------------------
[BF24/F4ED]  vgl. 9965
Zusatzpuffer von GCR nach Binär umwandeln
F8E0  A9 00      LDA #$00      Zeiger auf aktuelles GCR-Byte
F8E2  85 34      STA $34       zurücksetzen
F8E4  85 2E      STA $2E       Zeiger auf Zielpuffer löschen
F8E6  85 36      STA $36       Zeiger auf akt. Datenposition
F8E8  A9 01      LDA #$01      Zeiger $4E/$4F auf den
F8EA  85 4E      STA $4E       Beginn des Zusatzpuffers
F8EC  A9 BA      LDA #$BA      von $01BB-$01FF
F8EE  85 4F      STA $4F       setzen
F8F0  A5 31      LDA $31       Pufferzeiger auf Wert des aktuellen
F8F2  85 2F      STA $2F       Datenpuffers setzen
F8F4  20 E6 F7   JSR $F7E6     5 GCR-Bytes in 4 Binärbytes umwandeln
F8F7  A5 52      LDA $52       erstes umgewandeltes Byte
F8F9  85 38      STA $38       als Headerblockkennzeichen sezten
F8FB  A4 36      LDY $36       Zeiger in Puffer setzen
F8FD  A5 53      LDA $53       zweites umgewandeltes Byte
F8FF  91 2E      STA ($2E),Y   in aktuellen Datenpuffer schreiben
F901  C8         INY           drittes
F902  A5 54      LDA $54       umgewandeltes Byte
F904  91 2E      STA ($2E),Y   in aktuellen Datenpuffer schreiben
F906  C8         INY           viertes
F907  A5 55      LDA $55       umgewwandeltes Byte
F909  91 2E      STA ($2E),Y   in aktuellen Datenpuffer schreiben
F90B  C8         INY           Pufferzeiger auf nächstes Byte
F90C  84 36      STY $36       setzen und merken
F90E  20 E6 F7   JSR $F7E6     5 GCR-Bytes in 4 Binärbytes umwandeln
F911  A4 36      LDY $36       Pufferzeiger wieder holen
F913  A5 52      LDA $52       erstes umgewandeltes Byte
F915  91 2E      STA ($2E),Y   in aktuellen Datenpuffer schreiben
F917  C8         INY           Pufferzeiger auf nächstes Byte setzen
F918  F0 11      BEQ $F92B     Datenpuffer voll ?
F91A  A5 53      LDA $53       nein, zweites umgewandeltes Byte
F91C  91 2E      STA ($2E),Y   in aktuellen Datenpuffer schreiben
F91E  C8         INY           drittes umgewandeltes
F91F  A5 54      LDA $54       Byte in den
```

```
F921  91 2E     STA ($2E),Y    aktuellen Datenpuffer schreiben
F923  C8        INY            viertes umgewandeltes
F924  A5 55     LDA $55        Byte in den
F926  91 2E     STA ($2E),Y    aktuellen Datenpuffer schreiben
F928  C8        INY            Pufferzeiger auf nächstes Byte setzen
F929  D0 E1     BNE $F90C      Datenpuffer bereits voll ?
F92B  A5 53     LDA $53        ja, dann zweites umgewandeltes
F92D  85 3A     STA $3A        Byte als Prüfsumme (Parity) merken
F92F  A5 2F     LDA $2F        Zeiger auf aktuellen Datenpuffer
F931  85 31     STA $31        wieder herstellen
F933  60        RTS            zurück zur aufrufenden Routine
-----------------------------------------------------------------------
```

[972E/BF1B/F533]

Sektorheader in GCR-Bytes umwandeln

```
F934  A5 31     LDA $31        Zeiger auf aktuellen Datenpuffer
F936  85 2F     STA $2F        übernehmen
F938  A9 00     LDA #$00       Datenzeiger auf Headerpuffer
F93A  85 31     STA $31        richten
F93C  A9 24     LDA #$24       der bei $24
F93E  85 34     STA $34        beginnt
F940  A5 39     LDA $39        Kennzeichen für Blockheader (8)
F942  85 52     STA $52        in Zwischenspeicher für GCR-Routine
F944  A5 1A     LDA $1A        Prüfsumme des Blockheaders
F946  85 53     STA $53        in Zwischenspeicher für GCR-Routine
F948  A5 19     LDA $19        Sektornummer des Datenblocks
F94A  85 54     STA $54        in Zwischenspeicher für GCR-Routine
F94C  A5 18     LDA $18        Spurnummer des Datenblocks
F94E  85 55     STA $55        in Zwischenspeicher für GCR-Routine
F950  20 D0 F6  JSR $F6D0      4 Binärbytes in 5 GCR-Bytes umwandeln
F953  A5 17     LDA $17        zweites Zeichen der ID
F955  85 52     STA $52        in Zwischenspeicher für GCR-Routine
F957  A5 16     LDA $16        erstes Zeichen der ID
F959  85 53     STA $53        in Zwischenspeicher für GCR-Routine
F95B  A9 00     LDA #$00       Zwischenspeicher für GCR-Routine
F95D  85 54     STA $54        mit zwei Leerbytes
F95F  85 55     STA $55        auffüllen
F961  20 D0 F6  JSR $F6D0      4 Binärbytes in 5 GCR-Bytes umwandeln
F964  A5 2F     LDA $2F        aktuellen Datenpufferzeiger
F966  85 31     STA $31        wieder herstellen
F968  60        RTS            zurück zur aufrufenden Routine
-----------------------------------------------------------------------
```

```
[BF12/F2D5/F390/F40D/F420/F4F8/F507/F553/F583/F6C7/FDA0/FDE2]  vgl. 99B5
aktuellen Job beenden; Fehlerrückmeldung bereitstellen
F969  A4 3F      LDY $3F        Puffernummer des Jobs
F96B  99 00 00   STA $0000,Y    Fehlerrückmeldung in Befehlsregister
F96E  A5 50      LDA $50        Flag für GCR-Format
F970  F0 03      BEQ $F975      Daten noch im GCR-Code ?
F972  20 F2 F5   JSR $F5F2      ja, GCR-Daten umwandeln
F975  20 8F F9   JSR $F98F      Laufwerksmotor ausschalten
F978  A6 49      LDX $49        Zwischenspeicher für Stack
F97A  9A         TXS            Stack wieder zurücksetzen
F97B  4C BE F2   JMP $F2BE      zur Abfrage, ob neuer Job vorliegt
-----------------------------------------------------------------------
[92F5/F2DF]
Laufwerksmotor einschalten; warten bis Motor konstant läuft
F97E  A9 A0      LDA #$A0       Flag für 'Motor läuft an'
F980  85 20      STA $20        als Laufwerksstatus setzen
F982  AD 00 1C   LDA $1C00      Steuerregister holen
F985  09 04      ORA #$04       und Motorbit (Bit2)
F987  8D 00 1C   STA $1C00      auf 'Motor an' (=1) setzen
F98A  A9 32      LDA #$32       Zähler für 0.8 / 0.4 sek.
F98C  85 48      STA $48        Verzögerung setzen
F98E  60         RTS            zurück zur aufrufenden Routine
-----------------------------------------------------------------------
[86CF/98C1/F975]
Laufwerksmotor ausschalten
F98F  A6 3E      LDX $3E        aktuelles Laufwerk
F991  A5 20      LDA $20        Laufwerksstatus holen und
F993  09 10      ORA #$10       Flag für Motor ausschalten
F995  85 20      STA $20        setzen
F997  4C 2B A6   JMP $A62B      Zähler für Nachlaufzeit setzen
-----------------------------------------------------------------------
F99A  EA         NOP            unbenutztbenutzt  [entsteht durch
F99B  EA         NOP            Modifizierung des 1541-ROM]
-----------------------------------------------------------------------
[BF6F/F2E6/F2F0/F2F6/F31D/F48A/FAF2/FAFD/FD93/FDD8]
Hauptsteuerroutine des Diskkontrollers
F99C  AD 07 1C   LDA $1C07      Timer mit Anzahl der Taktzyklen
F99F  8D 05 1C   STA $1C05      (High-Byte) bis zum nächsten IRQ setzen
F9A2  AD 00 1C   LDA $1C00      Prüft Zustand der Lichtschranke für
F9A5  29 10      AND #$10       Schreibschutz um Diskettenwechsel
F9A7  C5 1E      CMP $1E        festzustellen
```

```
F9A9  85 1E      STA $1E         Schreibschutzstatus merken
F9AB  4C 34 A6   JMP $A634       Motor anschalten
-----------------------------------------------------------------
F9AE  EA         NOP             unbenutzter
F9AF  EA         NOP             Teil , der durch Modifizierung
F9B0  EA         NOP             des 1541-ROMS entstand
-----------------------------------------------------------------
[A657]
Kopf steuern
F9B1  AD FE 02   LDA $02FE       Flag für Zustand des Steppermotors
F9B4  F0 15      BEQ $F9CB       ist der Kopf auf der angewählten Spur ?
F9B6  C9 02      CMP #$02        nein, ist der Kopf gerade auf
F9B8  D0 07      BNE $F9C1       die angewählte Spur positioniert worden?
F9BA  A9 00      LDA #$00        ja, Flag für 'Kopf auf Spur'
F9BC  8D FE 02   STA $02FE       setzen
F9BF  F0 0A      BEQ $F9CB       immer Sprung nach $F9CB
F9C1  85 4A      STA $4A         Zähler für Anzahl der Halbschritte
F9C3  A9 02      LDA #$02        Flag für Kopfstatus
F9C5  8D FE 02   STA $02FE       auf 'Kopf auf Spur' setzen und einen
F9C8  4C 2E FA   JMP $FA2E       Halbschritt weiter auf angewählte Spur
F9CB  A6 3E      LDX $3E         Zustand des Laufwerksmotors
F9CD  30 07      BMI $F9D6       Läuft Motor ?
F9CF  A5 20      LDA $20         ja, prüfe Flag für Laufwerksstatus
F9D1  A8         TAY             merken
F9D2  C9 20      CMP #$20        Motor an und auf
F9D4  D0 03      BNE $F9D9       konstanter Drehzahl ?
F9D6  4C BE FA   JMP $FABE       Elektronik initialisieren
F9D9  C6 48      DEC $48         Verzögerungszähler für Anlaufen des
F9DB  D0 1D      BNE $F9FA       Motors; Motor auf Drehzahl ?
F9DD  98         TYA             ja, Drivestatus wieder holen
F9DE  10 04      BPL $F9E4       Laufwerk bereit ?
F9E0  29 7F      AND #$7F        nein, Flag für Motor-Anlaufphase
F9E2  85 20      STA $20         löschen und merken
F9E4  29 10      AND #$10        Bit für 'Motor läuft aus' testen
F9E6  F0 12      BEQ $F9FA       aktiv ?
F9E8  C6 35      DEC $35         ja, Zahl der Jobschleifenaufrufe
F9EA  D0 0E      BNE $F9FA       soll Stepper nochmal bediehnt werden ?
F9EC  EA         NOP             nein, Laufwerksmotor
F9ED  20 70 87   JSR $8770       ausschalten
F9F0  A9 FF      LDA #$FF        Flag für Laufwerk aktiv
```

```
F9F2  85 3E      STA $3E         löschen
F9F4  A9 00      LDA #$00        Flag für Laufwerksstatus
F9F6  85 20      STA $20         löschen
F9F8  F0 DC      BEQ $F9D6       immer Sprung nach $F9D6
F9FA  98         TYA             Flag für Laufwerksstatus
F9FB  29 40      AND #$40        Flag für Stepperzustand isolieren
F9FD  D0 03      BNE $FA02       Soll Kopf bewegt werden ?
F9FF  4C BE FA   JMP $FABE       nein,
FA02  6C 62 00   JMP ($0062)     ja, zur aktuellen Stepperroutine
```

mögliche Routinenaufrufe :

```
Kopfsteuerung initialisieren       $FA05
langsame Kopfbewegung              $FA3B
Kopfbewegung beenden               $FA4E
schnelle Bewegung einleiten        $FA7B
schnelle Kopfbewegung              $FA97
Kopf aus Schnellmodus abbremsen    $FAA5
```

```
[Einsprung über FA02]
Initialisierungsroutine für Kopfbewegung
FA05  A5 4A      LDA $4A         Zahl der Halbschritte bis zur Spur
FA07  10 05      BPL $FA0E       Soll der Kopf nach innen bewegt werden ?
FA09  49 FF      EOR #$FF        ja, dann Anzahl
FA0B  18         CLC             der Stepperschritte
FA0C  69 01      ADC #$01        mit positivem Vorzeichen versehen
FA0E  C5 64      CMP $64         auf Wert für 'Kopf auf Spur0' testen
FA10  B0 0A      BCS $FA1C       schnelle Kopfbewegung ?
FA12  A9 3B      LDA #$3B        nein, Aufruf in $FA02 auf
FA14  85 62      STA $62         Routine für langsame Kopfbewegung
FA16  A9 FA      LDA #$FA        setzen, indem Zeiger
FA18  85 63      STA $63         $62/$63 auf $FA3B gerichtet wird
FA1A  D0 12      BNE $FA2E       immer Sprung nach $FA2E
FA1C  E5 5E      SBC $5E         Anzahl der Schritte für Motor anfahren
FA1E  E5 5E      SBC $5E         und für Motor abbremsen (je 4) von den
FA20  85 61      STA $61         Gesamtschritten abziehen und merken
FA22  A5 5E      LDA $5E         Zeiger für Anzahl der Anfahrschritte
FA24  85 60      STA $60         setzen
FA26  A9 7B      LDA #$7B        Aufruf in $FA02 auf Routine
FA28  85 62      STA $62         für schnelle Kopfbewegung
```

```
FA2A  A9 FA     LDA #$FA      setzen, indem der Zeiger
FA2C  85 63     STA $63       $62/$63 auf $FA7B gerichtet wird
FA2E  A5 4A     LDA $4A       Schrittzähler
FA30  10 31     BPL $FA63     Bewegung nach innen ?
FA32  4C 36 FF  JMP $FF36     ja, Steppermotor ansteuern
----------------------------------------------------------------------
FA35  EA ...    NOP           unbenutzter
FA37  ... EA    NOP           ROM - Bereich
----------------------------------------------------------------------
[FF7F]
FA38  4C 69 FA  JMP $FA69     Stepper ansteuern
----------------------------------------------------------------------
[Einsprung über FA02]
Langsame Kopfbewegung für kurze Distanzen durchführen
FA3B  A5 4A     LDA $4A       Schrittzähler für Zahl der Halbschritte
FA3D  D0 EF     BNE $FA2E     Zielspur erreicht ?
FA3F  A9 4E     LDA #$4E      Aufruf in $FA02 auf Routine
FA41  85 62     STA $62       zum Beenden der Kopfbewegung
FA43  A9 FA     LDA #$FA      setzen, indem der Zeiger
FA45  85 63     STA $63       $62/$63 auf $FA7B gerichtet wird
FA47  A9 05     LDA #$05      Fünf Halbschritte zum Abstoppen
FA49  85 60     STA $60       des Kopfes setzen
FA4B  4C BE FA  JMP $FABE     Byte Ready Flag bereitmachen
----------------------------------------------------------------------
[Einsprung über FA02]
Beenden der Kopfbewegung
FA4E  C6 60     DEC $60       Zahl der Schritte zum Kopf abbremsen
FA50  D0 6C     BNE $FABE     Bremsvorgang ausgeführt ?
FA52  A5 20     LDA $20       Flag für Laufwerksstatus
FA54  29 BF     AND #$BF      Bitflag für Kopf in Bewegung
FA56  85 20     STA $20       zurücksetzen
FA58  A9 05     LDA #$05      Aufruf in $FA02 auf Routine
FA5A  85 62     STA $62       zur Initialisierung für Kopfbewegung
FA5C  A9 FA     LDA #$FA      setzen, indem der Zeiger
FA5E  85 63     STA $63       $62/$63 auf $FA05 gerichtet wird
FA60  4C BE FA  JMP $FABE     Byte Ready Flag bereitmachen
----------------------------------------------------------------------
[FA30]
Steppermotor steuern
FA63  C6 4A     DEC $4A       Zahl der Halbspurschritte
FA65  AE 00 1C  LDX $1C00     Steuerport für Steppermotor
```

```
FA68  E8        INX         Kopf nach außen bewegen indem die
FA69  8A        TXA         Stepperbits 0 und 1 aufwährts gezählt
FA6A  29 03     AND #$03    werden; Steuerbits isolieren
FA6C  85 4B     STA $4B     und merken
FA6E  AD 00 1C  LDA $1C00   Laufwerk-Steuerregister holen und
FA71  29 FC     AND #$FC    Steppermotorbits löschen
FA73  05 4B     ORA $4B     vorher berechnete Bits einblenden
FA75  8D 00 1C  STA $1C00   und Steppermotor ansteuern
FA78  4C BE FA  JMP $FABE   Byte Ready Flag bereitmachen
-----------------------------------------------------------------
[Einsprung über FA02]
Schnelle Kopfbewegung vorbereiten und Kopf anfahren
FA7B  38        SEC         Zeitkonstante
FA7C  AD 07 1C  LDA $1C07   bis zum nächsten Aufruf
FA7F  E5 5F     SBC $5F     um Anfahrkonstante (4) erniedrigen,
FA81  8D 05 1C  STA $1C05   um schnellere Stepperimpulse zum
FA84  C6 60     DEC $60     Anfahren zu liefern
FA86  D0 0C     BNE $FA94   Bereits vier Anfahrimpulse gegeben ?
FA88  A5 5E     LDA $5E     ja, Zähler für späteres Abremsen
FA8A  85 60     STA $60     setzen
FA8C  A9 97     LDA #$97    Aufruf in $FA02 auf Routine
FA8E  85 62     STA $62     zum schnellen Kopfbewegen
FA90  A9 FA     LDA #$FA    setzen, indem der Zeiger
FA92  85 63     STA $63     $62/$63 auf $FA97 gesetzt wird
FA94  4C 2E FA  JMP $FA2E   Kopf bewegen
-----------------------------------------------------------------
[Einsprung über FA02]
Schnelle Kopfbewegung durchführen
FA97  C6 61     DEC $61     Zähler für Halbschritte
FA99  D0 F9     BNE $FA94   am Ziel angelangt ?
FA9B  A9 A5     LDA #$A5    ja, Aufruf in $FA02 auf Routine
FA9D  85 62     STA $62     zum Abbremsen des Kopfs
FA9F  A9 FA     LDA #$FA    setzen, indem Zeiger
FAA1  85 63     STA $63     $62/$63 auf $FAA5 gesetzt wird
FAA3  D0 EF     BNE $FA94   immer Srung nach $FA94
-----------------------------------------------------------------
[Einsprung über FA02]
Abbremsen des Kopf nach schneller Bewegung
FAA5  AD 07 1C  LDA $1C07   Zeitkonstante bis zum nächsten Aufruf
FAA8  18        CLC         vergrößern, damit die Stepperimpulse
FAA9  65 5F     ADC $5F     zum Abbremsen langsamer werden, um ein
```

```
FAAB  8D 05 1C   STA $1C05      'Überlaufen' der Spur zu verhindern
FAAE  C6 60      DEC $60        Zähler für Bremsimpulse
FAB0  D0 E2      BNE $FA94      fertig abgebremst ?
FAB2  A9 4E      LDA #$4E       ja, Aufruf in $FA02 auf Routine
FAB4  85 62      STA $62        zum Beenden des Kofptransport
FAB6  A9 FA      LDA #$FA       setzen, indem der Zeiger
FAB8  85 63      STA $63        $62/$63 auf $FA4E gerichtet wird
FABA  A9 05      LDA #$05       Anzahl der Bremsimpulse
FABC  85 60      STA $60        neu setzen
-----------------------------------------------------------------
[F9D6/F9FF/FA4B/FA50/FA60/FA78/FF74]
FABE  AD 0C 1C   LDA $1C0C      Schreib-/Leseelektronik initialisieren
FAC1  29 FD      AND #$FD       indem Bit 1 (Byte Ready Flag)
FAC3  8D 0C 1C   STA $1C0C      zurückgesetzt wird
FAC6  60         RTS            zurück zur aufrufenden Routine
-----------------------------------------------------------------
Diskette im 1541-Format formatieren
Der Einsprung erfolgt über ($0600), wodurch man auch eigene
Vorprogramme einbauen kann
FAC7  A5 51      LDA $51        Nummer der aktuellen Spur
FAC9  10 2A      BPL $FAF5      Formatiervorgang bereits gestartet ?
FACB  A6 3D      LDX $3D        nein, Nummer des aktuellen Laufwerks
FACD  A9 60      LDA #$60       holen und in Flag für Laufwerksstatus
FACF  95 20      STA $20,X      Flag für Kopfbewegung setzem (Bit 6/5)
FAD1  A9 01      LDA #$01       Spur 1 als Startspur an Diskkontroller
FAD3  95 22      STA $22,X      übergeben und
FAD5  85 51      STA $51        merken
FAD7  A9 A4      LDA #$A4       Kopf 46 Spuren (bis zum Anschlag) nach
FAD9  85 4A      STA $4A        außen bewegen
FADB  AD 00 1C   LDA $1C00      Steuerbits für Steppermotor
FADE  29 FC      AND #$FC       löschen und an Stepper
FAE0  8D 00 1C   STA $1C00      übergeben
FAE3  A9 0A      LDA #$0A       max. Zahl der Formatierversuche
FAE5  8D 20 06   STA $0620      setzen
FAE8  A9 A0      LDA #$A0       Startwert für angenommene Spurkapazität
FAEA  8D 21 06   STA $0621      in $0621/$0622 auf $0FA0
FAED  A9 0F      LDA #$0F       gleich 4000 Bytes Kapazität
FAEF  8D 22 06   STA $0622      setzen
FAF2  4C 9C F9   JMP $F99C      Kopf auf Spur bewegen
FAF5  A0 00      LDY #$00       aktuelle Spurnummer mit
FAF7  D1 32      CMP ($32),Y    zwischengespeichertem Wert vergleichen
```

```
FAF9    F0 05       BEQ $FB00       noch die gleiche Spur in Bearbeitung ?
FAFB    91 32       STA ($32),Y     nein, aktuelle Spurnummer merken
FAFD    4C 9C F9    JMP $F99C       Kopf auf neue Spur bewegen
FB00[1] AD 00 1C    LDA $1C00       Steuerregister holen
FB03    29 10       AND #$10        und Write Protect (Bit4) prüfen
FB05    D0 05       BNE $FB0C       Schreibschutz vorhanden ?
FB07    A9 08       LDA #$08        ja, Fehlermeldung
FB09    4C D3 FD    JMP $FDD3       '26 Write Protect On' ausgeben
FB0C[2] 20 A3 FD    JSR $FDA3       gesamte Spur mit $FF beschreiben
FB0F    20 C3 FD    JSR $FDC3       Erwartete Spurkapazität mit $FF füllen
FB12    A9 55       LDA #$55        und dann mit der gleichen Anzahl
FB14    8D 01 1C    STA $1C01       $55-Bytes beschreiben
FB17    20 C3 FD    JSR $FDC3       die Kapazität ist in $0621/$622 vermerkt
FB1A    20 00 FE    JSR $FE00       Kopf auf Lesebetrieb umschalten
FB1D    20 56 F5    JSR $F556       auf erste $FF Bytes (Sync) warten
FB20    A9 40       LDA #$40        Bei Ablauf von Timer 1
FB22    0D 0B 18    ORA $180B       wird an PB7 (ATN-Eingang)
FB25    8D 0B 18    STA $180B       ein Impuls erzeugt
FB28    A9 62       LDA #$62        Zeitgeber 1 auf
FB2A    8D 06 18    STA $1806       eine Ablaufzeit von
FB2D    A9 00       LDA #$00        62 Taktimpulsen
FB2F    8D 07 18    STA $1807       programmieren
FB32    8D 05 18    STA $1805       Timer 1 starten
FB35    A0 00       LDY #$00        Zähler
FB37    A2 00       LDX #$00        löschen
FB39[1] 2C 00 1C    BIT $1C00       Sync-Flag testen
FB3C    30 FB       BMI $FB39       warten bis Sync-Signal vorbei ist
FB3E[1] 2C 00 1C    BIT $1C00       Sync-Flag überprüfen
FB41    10 FB       BPL $FB3E       warten bis Sync-Bereich wieder kommt
FB43[2] AD 04 18    LDA $1804       aktuellen Zählerstand von Timer 1 holen
FB46[1] 2C 00 1C    BIT $1C00       Sync-Flag testen
FB49    10 11       BPL $FB5C       ist Syncbereich jetzt vorbei ?
FB4B    AD 0D 18    LDA $180D       nein, Interruptflags holen
FB4E    0A          ASL A           und Flag für 'Timer 1 abgelaufen' testen
FB4F    10 F5       BPL $FB46       ist die Zeit zu Ende ?
FB51    E8          INX             ja, Zähler erhöhen
FB52    D0 EF       BNE $FB43       ist ein Übertrag entstanden ?
FB54    C8          INY             ja, High-Byte des Zählers korregieren
FB55    D0 EC       BNE $FB43       ist Zähler übergelaufen ?
FB57    A9 02       LDA #$02        ja, Fehlermeldung
FB59    4C D3 FD    JMP $FDD3       '20 Read Error' ausgeben
```

```
FB5C[1] 86 71     STX $71     Anzahl der $55-Bytes
FB5E    84 72     STY $72     merken
FB60    A2 00     LDX #$00    Register für nächstes Zählen
FB62    A0 00     LDY #$00    löschen
FB64[2] AD 04 18  LDA $1804   Zählerstand von Timer 1 holen
FB67[1] 2C 00 1C  BIT $1C00   Sync-Flag prüfen
FB6A    30 11     BMI $FB7D   ist Kopf über Sync Bereich ?
FB6C    AD 0D 18  LDA $180D   nein, Interruptflags holen
FB6F    0A        ASL A       und Flag für 'Timer 1 abgelaufen' testen
FB70    10 F5     BPL $FB67   ist die Zeit zu Ende ?
FB72    E8        INX         ja, Zähler erhöhen
FB73    D0 EF     BNE $FB64   ist ein Übertrag entstanden ?
FB75    C8        INY         ja, High-Byte des Zählers korregieren
FB76    D0 EC     BNE $FB64   ist Zähler übergelaufen ?
FB78    A9 02     LDA #$02    ja, Fehlermeldung
FB7A    4C D3 FD  JMP $FDD3   '20 Read Error' ausgeben
FB7D[1] 38        SEC         Differenz zwischen
FB7E    8A        TXA         dem $55 Bereich und
FB7F    E5 71     SBC $71     dem $FF Bereich
FB81    AA        TAX         berechnen und
FB82    85 70     STA $70     in Zeiger $70/$71 zur
FB84    98        TYA         Bestimmung der reellen
FB85    E5 72     SBC $72     Spurkapazität
FB87    A8        TAY         vermerken
FB88    85 71     STA $71     ($71/72 von X/Y abziehen und in $70/71)
FB8A    10 0B     BPL $FB97   ist Wert negativ ?
FB8C    49 FF     EOR #$FF    ja, 2-er Komplement des Wertes
FB8E    A8        TAY         bilden
FB8F    8A        TXA         (ergibt Absolutwert)
FB90    49 FF     EOR #$FF    Low-Byte komplementieren
FB92    AA        TAX         und merken
FB93    E8        INX         2-er Komplement bilden
FB94    D0 01     BNE $FB97   ist ein ein Übertrag entstanden ?
FB96    C8        INY         ja, High-Byte korregieren
FB97[2] 98        TYA         und holen
FB98    D0 04     BNE $FB9E   ist Wert in X/Y kleiner 256 ?
FB9A    E0 04     CPX #$04    ja, Low-Byte (X) mit 4 vergleichen
FB9C    90 18     BCC $FBB6   Kapazität der Spur auf 4 Bytes genau ?
FB9E[1] 06 70     ASL $70     nein, Wert der Spurkapazität
FBA0    26 71     ROL $71     verdoppeln
FBA2    18        CLC         und erwartete Spurkapazität dazurechnen
```

```
FBA3     A5 70      LDA $70      Low-Byte holen und
FBA5     6D 21 06   ADC $0621    erwarteten Wert addieren
FBA8     8D 21 06   STA $0621    neuen erwarteten Wert merken
FBAB     A5 71      LDA $71      High-Byte holen
FBAD     6D 22 06   ADC $0622    und zu erwartetem Wert
FBB0     8D 22 06   STA $0622    dazurechnen
FBB3     4C 0C FB   JMP $FB0C    Spurkapazitat nochmal bestimmen
FBB6[1]  A2 00      LDX #$00     Zähler
FBB8     A0 00      LDY #$00     löschen
FBBA     B8         CLV          Flag fur 'Byte Ready' bereit machen
FBBB[3]  AD 00 1C   LDA $1C00    Flag für Sync-Signal testen
FBBE     10 0E      BPL $FBCE    ist Kopf über Sync-Bereich ?
FBC0     50 F9      BVC $FBBB    ja, auf nächstes Byte warten
FBC2     B8         CLV          Byte Ready wieder bereit machen
FBC3     E8         INX          Zähler erhöhen
FBC4     D0 F5      BNE $FBBB    ist ein Übertrag entstanden ?
FBC6     C8         INY          ja, Highbyte des Zählers korregieren
FBC7     D0 F2      BNE $FBBB    ist Zähler übergelaufen
FBC9     A9 03      LDA #$03     ja, Fehlernummer für 'Sync not found'
FBCB     4C D3 FD   JMP $FDD3    setzen; eventuell weiteren Versuch
FBCE[1]  8A         TXA          Zähler verdoppeln und in $0625/$0624 :
FBCF     0A         ASL A        Low-Byte verdoppeln
FBD0     8D 25 06   STA $0625    und merken
FBD3     98         TYA          High-Byte holen
FBD4     2A         ROL A        und mal zwei nehmen
FBD5     8D 24 06   STA $0624    Wert merken
FBD8     A9 BF      LDA #$BF     Flag für 'Ablauf von Timer 1'
FBDA     2D 0B 18   AND $180B    Interruptflags holen
FBDD     8D 0B 18   STA $180B    und Flag zurücksetzen
FBE0     A9 66      LDA #$66     Zahl der Bytes die jeder Sektor
FBE2     8D 26 06   STA $0626    zusätzlich zu den 256 Daten benötigt
FBE5     A6 43      LDX $43      Zahl der Sektoren
FBE7     A0 00      LDY #$00     Indexwert für Zahl der 256 Byte Blöcke
FBE9     98         TYA          Startwert für Überschußrechnung (0)
FBEA[1]  18         CLC          und dazu den Überschuß des Sektors
FBEB     6D 26 06   ADC $0626    anrechnen
FBEE     90 01      BCC $FBF1    sind weitere 256 Bytes nötig
FBF0     C8         INY          Index um 256 Bytes hochsetzen
FBF1[1]  C8         INY          Index um 256 Bytes hochsetzen
FBF2     CA         DEX          nächsten Sektor einrechnen
FBF3     D0 F5      BNE $FBEA    sind alle Sektoren berücksichtigt ?
```

```
FBF5  49 FF     EOR #$FF     ja, 2er Komplement (negativen Wert)
FBF7  38        SEC          der restlichen benötigten
FBF8  69 00     ADC #$00     Bytes errechen
FBFA  18        CLC          und von Gesamtkapazität
FBFB  6D 25 06  ADC $0625    abziehen (addiere negativen Wert)
FBFE  B0 03     BCS $FC03    ist ein Borgen nötig ?
FC00  CE 24 06  DEC $0624    ja, High-Byte korregieren
FC03  AA        TAX          Low-Byte der Kapazität merken
FC04  98        TYA          Zahl der nötigen 256 Byte-Blöcke holen
FC05  49 FF     EOR #$FF     und davon 2er Komplement
FC07  38        SEC          (negativer Wert)
FC08  69 00     ADC #$00     bilden
FC0A  18        CLC          Zahl der nötigen 256-Byte-Blocks
FC0B  6D 24 06  ADC $0624    von Gesamtkapazität abziehen
FC0E  10 05     BPL $FC15    reicht Spurkapazität aus ?
FC10  A9 04     LDA #$04     nein, Fehlermeldung 'Block Not Found'
FC12  4C D3 FD  JMP $FDD3    ausgeben
FC15  A8        TAY          Zahl der noch übrigen Bytes
FC16  8A        TXA          holen
FC17  A2 00     LDX #$00     Zähler für Zahl der Lückenbytes
FC19  38        SEC          Zahl der übrigen Bytes
FC1A  E5 43     SBC $43      durch Zahl der Sektoren teilen
FC1C  B0 03     BCS $FC21    indem immer die Sektorenzahl von den
FC1E  88        DEY          Lückenbytes subtrahiert wird
FC1F  30 03     BMI $FC24    X zählt dabei wie oft das möglich ist
FC21  E8        INX          Zahl der Lückennbytes erhöhen
FC22  D0 F5     BNE $FC19    immer Sprung nach $FC19
FC24  8E 26 06  STX $0626    Zahl der Lückenbytes merken
FC27  E0 04     CPX #$04     und mit 4 Bytes vergleichen
FC29  B0 05     BCS $FC30    ist Lücke kleiner ?
FC2B  A9 05     LDA #$05     ja, Fehlermeldung
FC2D  4C D3 FD  JMP $FDD3    '23 Read Error' ausgeben
FC30  18        CLC          Zahl der Sektoren
FC31  65 43     ADC $43      der Spur addieren
FC33  8D 27 06  STA $0627    und Wert merken
FC36  A9 00     LDA #$00     Zähler für
FC38  8D 28 06  STA $0628    geschriebene Sektoren zurücksetzen
FC3B  A0 00     LDY #$00     Zeiger für Blockheader erstellen in
FC3D  A6 3D     LDX $3D      Puffer 1 löschen
FC3F  A5 39     LDA $39      Kennzeichen für Blockheader (8)
FC41  99 00 03  STA $0300,Y  in Blockheader schreiben
```

```
FC44  C8        INY          Zeiger auf nächste Position setzen
FC45  C8        INY          Byte für Prüfsumme überspringen
FC46  AD 28 06  LDA $0628    Nummer des aktuellen Sektors
FC49  99 00 03  STA $0300,Y  in Blockheader schreiben
FC4C  C8        INY          Zeiger auf nächste Position setzen
FC4D  A5 51     LDA $51      Nummer der aktuellen Spur
FC4F  99 00 03  STA $0300,Y  in Blockheader übernehmen
FC52  C8        INY          Zeiger auf nächste Position setzen
FC53  B5 13     LDA $13,X    zweites Zeichen der ID
FC55  99 00 03  STA $0300,Y  in Blockheader schreiben
FC58  C8        INY          Zeiger auf nächste Position setzen
FC59  B5 12     LDA $12,X    erstes Zeichen der ID
FC5B  99 00 03  STA $0300,Y  in Blockheader übertragen
FC5E  C8        INY          Zeiger auf nächste Position setzen
FC5F  A9 0F     LDA #$0F     Zweimal $0F (15)
FC61  99 00 03  STA $0300,Y  zum Auffüllen des
FC64  C8        INY          Blockheaders in
FC65  99 00 03  STA $0300,Y  den Puffer
FC68  C8        INY          schreiben
FC69  A9 00     LDA #$00     Prüfsumme von :
FC6B  59 FA 02  EOR $02FA,Y  Spurnummer
FC6E  59 FB 02  EOR $02FB,Y  Sektornummer
FC71  59 FC 02  EOR $02FC,Y  zweites ID-Zeichen
FC74  59 FD 02  EOR $02FD,Y  erstes ID-Zeichen
FC77  99 F9 02  STA $02F9,Y  berechnen und im Blockheader setzen
FC7A  EE 28 06  INC $0628    Zähler für aktuelle  Sektornummer
FC7D  AD 28 06  LDA $0628    auf nächsten Sektor setzen und mit Wert
FC80  C5 43     CMP $43      der max. Sektornummer vergleichen
FC82  90 BB     BCC $FC3F    schon alle Sektoren angelegt ?
FC84  98        TYA          ja, Zeiger auf aktuelle
FC85  48        PHA          Pufferposition retten
FC86  E8        INX          (1)
FC87  8A        TXA          Datenblock herstellen
FC88[1] 9D 00 05  STA $0500,X  1 in Puffer schreiben
FC8B  E8        INX          Zeiger auf nächstes Byte setzen
FC8C  D0 FA     BNE $FC88    Puffer voll ?
FC8E  A9 03     LDA #$03     ja, Adresse $0300 als aktuelle
FC90  85 31     STA $31      Pufferadresse setzen
FC92  20 30 FE  JSR $FE30    Pufferinhalt in GCR-Code umwandeln
FC95  68        PLA          vorherige Pufferposition
FC96  A8        TAY          wiederherstellen und Zeiger auf
```

```
FC97     88         DEY            Beginn des Blockheaders setzen
FC98     20 E5 FD   JSR $FDE5      Inhalt des Zusatzpuffers in den
FC9B     20 F5 FD   JSR $FDF5      Puffer ab $0300 verschieben
FC9E     A9 05      LDA #$05       $0500 als aktuelle
FCA0     85 31      STA $31        Pufferadresse setzen
FCA2     20 E9 F5   JSR $F5E9      Prüfsumme des Datenblocks errechnen
FCA5     85 3A      STA $3A        und merken
FCA7     20 8F F7   JSR $F78F      Datenblock in GCR-Code umrechenen
FCAA     A9 00      LDA #$00       Zeiger auf aktuellen
FCAC     85 32      STA $32        Blockheader initialisieren
FCAE     20 0E FE   JSR $FE0E      Spur mit $55 löschen
FCB1[1]  A9 FF      LDA #$FF       Kennzeichen für Sync-Markierung
FCB3     8D 01 1C   STA $1C01      an Schreibkopf übergeben
FCB6     A2 05      LDX #$05       Anzahl der Sync-Bytes
FCB8[2]  50 FE      BVC $FCB8      Auf 'Byte Ready' warten
FCBA     B8         CLV            Byte Ready Flag wieder bereit machen
FCBB     CA         DEX            Zähler erniedrigen
FCBC     D0 FA      BNE $FCB8      schon alle Sync-Bytes auf Diskette ?
FCBE     A2 0A      LDX #$0A       Länge des Blockheaders
FCC0     A4 32      LDY $32        Zeiger auf Position im Puffer
FCC2[2]  50 FE      BVC $FCC2      Schreibelektronik bereit ?
FCC4     B8         CLV            ja, Flag wieder herstellen
FCC5     B9 00 03   LDA $0300,Y    GCR-Bytes aus Puffer holen
FCC8     8D 01 1C   STA $1C01      an Schreibkopf übertragen
FCCB     C8         INY            Pufferzeiger auf nächstes Zeichen
FCCC     CA         DEX            Zahl der noch zu schreibenden Zeichen
FCCD     D0 F3      BNE $FCC2      Header fertig geschrieben ?
FCCF     A2 09      LDX #$09       ja, Lücke zwischen Blockheader und
FCD1[2]  50 FE      BVC $FCD1      Datenblock mit neun
FCD3     B8         CLV            Füllwerten ($55)
FCD4     A9 55      LDA #$55       beschreiben
FCD6     8D 01 1C   STA $1C01      Byte an Schreibkopf übergeben
FCD9     CA         DEX            Zähler für Zahl der Füllbytes
FCDA     D0 F5      BNE $FCD1      Lücke fertig geschrieben ?
FCDC     A9 FF      LDA #$FF       Sync-Marke für den
FCDE     A2 05      LDX #$05       Datenblockheader auf  Diskette schreiben
FCE0[2]  50 FE      BVC $FCE0      Schreibelektronik bereit ?
FCE2     B8         CLV            ja, Flag wiederherstellen
FCE3     8D 01 1C   STA $1C01      Sync-Byte an Schreibelektronik
FCE6     CA         DEX            Zähler für Zahl der Sync-Bytes
FCE7     D0 F7      BNE $FCE0      Sync-Markierung fertig geschrieben ?
```

```
FCE9     A2 BB      LDX #$BB      Zeiger auf Beginn des Zwischenpuffers
FCEB[2]  50 FE      BVC $FCEB     Schreibelektronik bereit ?
FCED     B8         CLV           ja, Byte Ready Flag bereitmachen
FCEE     BD 00 01   LDA $0100,X   Byte aus Puffer holen
FCF1     8D 01 1C   STA $1C01     und auf Diskette schreiben
FCF4     E8         INX           Pufferzeiger auf nächstes Byte
FCF5     D0 F4      BNE $FCEB     Puffer geschrieben ?
FCF7     A0 00      LDY #$00      ja, Pufferzeiger auf Datenpuffer
FCF9[2]  50 FE      BVC $FCF9     Schreibelektronik bereit ?
FCFB     B8         CLV           ja, Byte Ready Flag bereit machen
FCFC     B1 30      LDA ($30),Y   Byte aus Puffer
FCFE     8D 01 1C   STA $1C01     auf Diskette schreiben
FD01     C8         INY           Zeiger auf nächstes Zeichen in Puffer
FD02     D0 F5      BNE $FCF9     schon ganzer Puffer geschrieben
FD04     A9 55      LDA #$55      Lücke zwischen zwei Datenblöcken füllen
FD06     AE 26 06   LDX $0626     Anzahl der Bytes pro Lücke
FD09[2]  50 FE      BVC $FD09     Schreibelektronik bereit ?
FD0B     B8         CLV           ja, Flag zurücksetzen
FD0C     8D 01 1C   STA $1C01     $55 zum Schreibkopf
FD0F     CA         DEX           Zähler für Anzahl der Füllbytes
FD10     D0 F7      BNE $FD09     Lücke fertig geschrieben ?
FD12     A5 32      LDA $32       Pufferzeiger auf Position des Headers
FD14     18         CLC           auf nächsten
FD15     69 0A      ADC #$0A      Blockheader setzen
FD17     85 32      STA $32       und merken
FD19     CE 28 06   DEC $0628     Nummer des nächsten Sektors bilden
FD1C     D0 93      BNE $FCB1     bereits alle Sektoren geschrieben ?
FD1E[1]  50 FE      BVC $FD1E     ja, auf nächstes Byte warten
FD20     B8         CLV           Byte Ready Flag bereit machen
FD21[1]  50 FE      BVC $FD21     aus nächstes Byte warten
FD23     B8         CLV           Byte Ready zurücksetzen
FD24     20 00 FE   JSR $FE00     auf Lesebetrieb umschalten
FD27     A9 C8      LDA #$C8      Zahl der Leseversuche (200)
FD29     8D 23 06   STA $0623     setzen
FD2C     A9 00      LDA #$00      Pufferzeiger $30/$31
FD2E     85 30      STA $30       auf Puffer 1
FD30     A9 03      LDA #$03      ($0300-$03FF)
FD32     85 31      STA $31       setzen
FD34     A5 43      LDA $43       Zahl der Sektoren pro Spur
FD36     8D 28 06   STA $0628     merken
FD39[1]  20 56 F5   JSR $F556     auf Sync-Markierung warten
```

```
FD3C  A2 0A     LDX #$0A      Anzahl der Bytes im Blockheader
FD3E  A0 00     LDY #$00      Pufferzeiger löschen
FD40  50 FE     BVC $FD40     Leseelektronik bereit ?
FD42  B8        CLV           ja, Flag wieder bereit machen
FD43  AD 01 1C  LDA $1C01     Byte von Diskette lesen
FD46  D1 30     CMP ($30),Y   und mit Puffer vergleichen
FD48  D0 0E     BNE $FD58     gesuchter Blockheader ?
FD4A  C8        INY           Zeiger auf nächstes Byte des Headers
FD4B  CA        DEX           setzen
FD4C  D0 F2     BNE $FD40     letztes Byte des Headers verglichen ?
FD4E  18        CLC           Pufferadresse auf
FD4F  A5 30     LDA $30       nächsten Blockheader
FD51  69 0A     ADC #$0A      im Pufferspeicher
FD53  85 30     STA $30       setzen
FD55  4C 62 FD  JMP $FD62     weiter
FD58  CE 23 06  DEC $0623     Zahl der Leseversuche
FD5B  D0 CF     BNE $FD2C     letzter Versuch ?
FD5D  A9 06     LDA #$06      ja, Fehlermeldung
FD5F  4C D3 FD  JMP $FDD3     '24 Read Error' ausgeben
FD62  20 56 F5  JSR $F556     auf Sync-Marke des Datenblocks warten
FD65  A0 BB     LDY #$BB      Pufferzeiger auf Zwischenpuffer setzen
FD67  50 FE     BVC $FD67     Leseelektronik bereit ?
FD69  B8        CLV           ja, Byte Ready Flag zurücksetzen
FD6A  AD 01 1C  LDA $1C01     Byte von Diskette
FD6D  D9 00 01  CMP $0100,Y   mit Pufferinhalt vergleichen
FD70  D0 E6     BNE $FD58     Vergleich positiv ?
FD72  C8        INY           ja, Pufferzeiger auf nächstes Byte
FD73  D0 F2     BNE $FD67     bereits ganzer Puffer verglichen ?
FD75  A2 FC     LDX #$FC      ja, Zähler für Bytes in Datenpuffer
FD77  50 FE     BVC $FD77     Leseelektronik bereit ?
FD79  B8        CLV           ja, Byte Ready Flag zurücksetzen
FD7A  AD 01 1C  LDA $1C01     Byte von Diskette lesen
FD7D  D9 00 05  CMP $0500,Y   und mit Datenpuffer vergleichen
FD80  D0 D6     BNE $FD58     Vergleich positiv ?
FD82  C8        INY           ja, Zeiger auf nächstes Byte
FD83  CA        DEX           setzen
FD84  D0 F1     BNE $FD77     letztes Zeichen des Puffers
FD86  CE 28 06  DEC $0628     Zahl der Sektoren -1 der Spur
FD89  D0 AE     BNE $FD39     alle Sektoren überprüft ?
FD8B  E6 51     INC $51       ja, Zähler für Spurnummer auf nächste
FD8D  A5 51     LDA $51       Spur setzen und merken und
```

```
FD8F   C9 24      CMP #$24      mit max. Spurnummer vergleichen
FD91   B0 03      BCS $FD96     Spur 35 erreicht ?
FD93   4C 9C F9   JMP $F99C     nein, weiter formatieren
FD96[1] A9 FF     LDA #$FF      Flag für Formatierung zu
FD98   85 51      STA $51       Ende  setzen
FD9A   A9 00      LDA #$00      Flag für 'Puffer in GCR-Code'
FD9C   85 50      STA $50       löschen
FD9E   A9 01      LDA #$01      Rückmeldung 'ok'
FDA0   4C 69 F9   JMP $F969     ausgeben und Ende der Formatierung
--------------------------------------------------------------------------------
[FB0C]
Spur mit $FF beschreiben
FDA3   AD 0C 1C   LDA $1C0C     Kopfelektronik
FDA6   29 1F      AND #$1F      in PCR-Register auf
FDA8   09 C0      ORA #$C0      Schreibbetrieb
FDAA   8D 0C 1C   STA $1C0C     umschalten (CB2 = 0)
FDAD   A9 FF      LDA #$FF      Port des Schreib-/Lesekopfes auf
FDAF   8D 03 1C   STA $1C03     Ausgabe schalten
FDB2   8D 01 1C   STA $1C01     $FF schreiben
FDB5   A2 28      LDX #$28      Zähler in CPU-Register
FDB7   A0 00      LDY #$00      auf 10240 setzen
FDB9[3] 50 FE     BVC $FDB9     auf Byte Ready warten
FDBB   B8         CLV           Byte Ready Flag wieder bereit machen
FDBC   88         DEY           Low-Byte Zähler
FDBD   D0 FA      BNE $FDB9     einmal bis Null durchgelaufen ?
FDBF   CA         DEX           ja, dann High-Byte Zähler erniedrigen
FDC0   D0 F7      BNE $FDB9     bereits 10240 mal geschrieben ?
FDC2   60         RTS           ja, zurück zur aufrufenden Routine
--------------------------------------------------------------------------------
[FB0F/FB17]
($0621/$0622) mal 'Byte Ready' Signal abwarten
FDC3   AE 21 06   LDX $0621     Schleifenzähler
FDC6   AC 22 06   LDY $0622     setzen
FDC9[3] 50 FE     BVC $FDC9     auf Byte Ready warten
FDCB   B8         CLV           Byte Ready Flag zurücksetzen
FDCC   CA         DEX           Low-Byte Zähler
FDCD   D0 FA      BNE $FDC9     auf Null ?
FDCF   88         DEY           ja, dann Y erniedrigen
FDD0   10 F7      BPL $FDC9     Y mal auf 256 Byte Ready gewartet ?
FDD2   60         RTS           ja, zurück zur aufrufenden Routine
--------------------------------------------------------------------------------
```

```
[FB09/FB59/FB7A/FBCB/FC12/FC2D/FD5F]
Steuert Verhalten bei Fehlern beim Formatieren
FDD3  CE 20 06   DEC $0620       Anzahl der Formatierversuche -1
FDD6  F0 03      BEQ $FDDB       Fehler beim Formatieren aufgetreten ?
FDD8  4C 9C F9   JMP $F99C       nein, dann weiter formatieren
FDDB  A0 FF      LDY #$FF        Flag für 'Formatieren zu Ende'
FDDD  84 51      STY $51         setzen
FDDF  C8         INY             Flag für 'Puffer in GCR-Code'
FDE0  84 50      STY $50         löschen
FDE2  4C 69 F9   JMP $F969       Formatieren Beenden
--------------------------------------------------------------------
[FC98/FDEC]
Kopiert Bytes in Puffer 0 um 70 Bytes weiter nach oben
(Y-Register muß die Anzahl der zu kopierenden Bytes enthalten)
FDE5  B9 00 03   LDA $0300,Y     Byte aus Pufferanfang holen
FDE8  99 45 03   STA $0345,Y     und nach oben übertragen
FDEB  88         DEY             nächstes Byte anwählen
FDEC  D0 F7      BNE $FDE5       bereits alle Bytes ?
FDEE  AD 00 03   LDA $0300       ja, dann letztes Byte
FDF1  8D 45 03   STA $0345       kopieren und
FDF4  60         RTS             zurück zur aufrufenden Routine
--------------------------------------------------------------------
[FC9B]
Kopiert den Bereich $01BB-$01FF in den Puffer auf den $30/$31 zeigt
FDF5  A0 44      LDY #$44        Startposition $01FF
FDF7  B9 BB 01   LDA $01BB,Y     Byte aus Zwischenpuffer holen
FDFA  91 30      STA ($30),Y     und in Datenpuffer übertragen
FDFC  88         DEY             nächstes Byte anwählen
FDFD  10 F8      BPL $FDF7       bereits alle Bytes übertragen ?
FDFF  60         RTS             ja, zurück zur aufrufenden Routine
--------------------------------------------------------------------
[8D59/9AE6/9CCC/FB1A/FD24/BF0C]
Kopfelektronik von Schreiben auf Lesen umschalten
FE00  AD 0C 1C   LDA $1C0C       Steuerregister holen
FE03  09 E0      ORA #$E0        und Kopf auf Lesen umschalten
FE05  8D 0C 1C   STA $1C0C       (CB2 Ausgang =1)
FE08  A9 00      LDA #$00        Datenport zum Kopf auf
FE0A  8D 03 1C   STA $1C03       Eingabe schalten
FE0D  60         RTS             zurück zur aufrufenden Routine
--------------------------------------------------------------------
```

```
[FCAE]
Gesamte Spur mit $55 beschreiben
FE0E  AD 0C 1C   LDA $1C0C      Steuerregister holen
FE11  29 1F      AND #$1F       und Kopf auf Schreiben umstellen
FE13  09 C0      ORA #$C0       Bit 5-7 ausblenden und Bit 6/7 setzen
FE15  8D 0C 1C   STA $1C0C      (CB2 Ausgang =0)
FE18  A9 FF      LDA #$FF       Datenport zum Kopf auf
FE1A  8D 03 1C   STA $1C03      Ausgabe schalten
FE1D  A9 55      LDA #$55       $55 an Schreibkopf
FE1F  8D 01 1C   STA $1C01      übergeben
FE22  A2 28      LDX #$28       Registerzähler auf
FE24  A0 00      LDY #$00       10240 mal Schreiben einstellen
FE26  50 FE      BVC $FE26      Elektronik für nächstes Byte bereit ?
FE28  B8         CLV            ja, Flag wieder zurücksetzen
FE29  88         DEY            256 Bytes schreiben
FE2A  D0 FA      BNE $FE26      bereits 256 Bytes ?
FE2C  CA         DEX            ja, 40 mal 256 Bytes schreiben
FE2D  D0 F7      BNE $FE26      bereits 40 mal geschrieben ?
FE2F  60         RTS            ja, zurück zur aufrufenden Routine
---------------------------------------------------------------------
[9BEC/FC92]
Blockheader von Binär in GCR umrechnen
FE30  A9 00      LDA #$00       Pufferzeiger auf Start
FE32  85 30      STA $30        zurücksetzen
FE34  85 2E      STA $2E        Low-Byte des Zeigers auf Binärdaten
FE36  85 36      STA $36        Position in aktuellem Puffer
FE38  A9 BB      LDA #$BB       Positionszeiger auf
FE3A  85 34      STA $34        Zusatzpuffer richten
FE3C  A5 31      LDA $31        Zeiger auf aktuellen Datenpuffer
FE3E  85 2F      STA $2F        übernehmen
FE40  A9 01      LDA #$01       Zeiger auf Zusatzpuffer
FE42  85 31      STA $31        setzen
FE44  A4 36      LDY $36        aktuelle Position feststellen
FE46  B1 2E      LDA ($2E),Y    Byte aus Puffer holen und
FE48  85 52      STA $52        als erstes umzuwandelndes Byte merken
FE4A  C8         INY            Zeiger auf nächstes Byte richten
FE4B  B1 2E      LDA ($2E),Y    Byte aus Puffer holen und
FE4D  85 53      STA $53        als zweites umzuwandelndes Byte merken
FE4F  C8         INY            Zeiger auf nächstes Byte richten
FE50  B1 2E      LDA ($2E),Y    Byte aus Puffer holen und
FE52  85 54      STA $54        als drittes umzuwandelndes Byte merken
```

```
FE54  C8         INY            Zeiger auf nächstes Byte richten
FE55  B1 2E      LDA ($2E),Y    Byte aus Puffer holen und
FE57  85 55      STA $55        als drittes umzuwandelndes Byte merken
FE59  C8         INY            Zeiger auf nächstes Byte richten
FE5A  F0 08      BEQ $FE64      Ende des Puffers erreicht ?
FE5C  84 36      STY $36        nein, Position merken
FE5E  20 D0 F6   JSR $F6D0      4 Binärbytes in 5 GCR-Bytes umrechnen
FE61  4C 44 FE   JMP $FE44      weiter umwandeln
FE64[1] 4C D0 F6 JMP $F6D0      4 Binärbytes in 5 GCR-Bytes umrechnen
-----------------------------------------------------------------------
[Einsprung über Systemvektor FFFE]
FE67  6C A9 02 JMP ($02A9)      Sprung zur IRQ-Routine $9D88/$9DDE
-----------------------------------------------------------------------
FE6A  FF ...                    unbenutzter
FE84  ... FF                    ROM-Bereich
-----------------------------------------------------------------------
Aufbau des Directory und der BAM
FE85  12                        Nummer der Directoryspur (18)
FE86  04                        Zahl der Bytes für jede Spur in der BAM
FE87  04                        Startposition der BAM in Sektor 18,0
FE88  90                        Beginn des Diskettennamens (Pos. 144)
-----------------------------------------------------------------------
Tabelle der Floppybefehle
FE89  56                        'V' :  Validate / Collect
FE8A  49                        'I' :  Initialize
FE8B  44                        'D' :  Duplicate (nur Doppellaufwerke)
FE8C  4D                        'M' :  Memory Befehl
FE8D  42                        'B' :  Block Befehl
FE8E  55                        'U' :  User Befehl
FE8F  50                        'P' :  Position / Record
FE90  26                        '&' :  & - Befehl
FE91  43                        'C' :  Copy
FE92  52                        'R' :  Rename
FE93  53                        'S' :  Scratch
FE94  4E                        'N' :  New / Header
-----------------------------------------------------------------------
Adressen der Floppy-Befehle
FE95  84 05 C1 F8 1B 5C 07 A3   Low-Bytes der Einspungadressen
FE9D  F0 88 23 0D               für die Befehle
FEA1  ED D0 C8 CA CC CB E2 E7   High-Bytes der Einprungadressen
FEA9  C8 CA C8 EE               für die Befehle
```

```
--------------------------------------------------------------------
Bitmuster zum Prüfen der Befehlssyntax

Bedeutung der Bits :
(1=wird geprüft; entsprechdes Bit im Prüfwert muß dann 0 sein)

Bit0  '=' Zeichen im Befehlsstring vorhanden
Bit1  nach '=' Zeichen weitere Parameter vorhanden
Bit2  mehrere Dateinamen für 2. Dateinennung
Bit3  Joker in 2. Dateinennung vorhanden
Bit6  mehrere Dateinamen für 1. Dateinennung
Bit7  Joker in 1. Dateinennung vorhanden

FEAD  51                         %01010001  Datei(en) kopieren
FEAE  DD                         %11011101  Datei umbenennen
FEAF  1C                         %00011100  Datei(en) löschen
FEB0  9E                         %10011110  Diskette Formatieren
FEB1  1C                         %00011100  Datei lesen
--------------------------------------------------------------------
Kennzeichen im Befehlsstring für Betriebsmodus
FEB2  52 57 41 4D                R , W , A , M
--------------------------------------------------------------------
Dateitypkennzeichen in Befehlsstring
FEB6  44 53 50 55 4C             D , S , P , U , L
--------------------------------------------------------------------
Namen der verschiedenen Dateitypen
FEBB  44 53 50 55 52             1. Buchstabe :  D ,  S ,  P ,  U ,  R
FEC0  45 45 52 53 45             2. Buchstabe :  E ,  E ,  R ,  S ,  E
FEC5  4C 51 47 52 4C             3. Buchstabe :  L ,  Q ,  G ,  R ,  L
--------------------------------------------------------------------
Maske für LED-Bit im Steuerregister
FECA  08 00                      Laufwerk 0, Laufwerk 1 (nicht vorhanden)
--------------------------------------------------------------------
Flags um Prozessorstatus zu setzen
FECC  00                         N=0    V=0    Z=1
FECD  3F                         N=0    V=0    Z=0
FECE  7F                         N=0    V=1    Z=0
FECF  BF                         N=1    V=0    Z=0
FED0  FF                         N=1    V=1    Z=0
--------------------------------------------------------------------
```

```
Zahl der Sektoren in bestimmtem Spurbereich
FED1  11                            Spur 31-35 :  17 Sektoren
FED2  12                            Spur 25-30 :  18 Sektoren
FED3  13                            Spur 18-24 :  19 Sektoren
FED4  15                            Spur 01-17 :  21 Sektoren
------------------------------------------------------------------
FED5  41                            Kennzeichen für 1541-Format ('A')
------------------------------------------------------------------
FED6  04                            Anzahl der Spurwechsel
------------------------------------------------------------------
Spuren, bei dehnen die Sektorenzahl und Bitrate geändert wird
FED7  24 1F 19 12                   Spurnummer 36, 31, 25 und 18
------------------------------------------------------------------
FEDB  01 FF FF 01 00                Steuerbytes für Kopf bei Lesefehlern
------------------------------------------------------------------
Lage der Puffer im Speicher
FEE0  03 04 05 06 07 07             High-Bytes der Pufferadressen
------------------------------------------------------------------
FEE6  FF                            Leerbyte (bei 1541 DOS Prüfsumme)
------------------------------------------------------------------
[FF0B]  Reset ohne Harwaretest; Zeiger gesetzt durch $EBC5
FEE7  6C 65 00   JMP ($0065)        Sprung nach $EB22
------------------------------------------------------------------
[EA7A]
LED initialisieren und anschalten
FEEA  8D 00 1C   STA $1C00          Bit für 'LED an' setzen (8)
FEED  8D 02 1C   STA $1C02          und Pin auf Ausgang schalten
FEF0  4C 7D EA   JMP $EA7D          zurück zur Harwarefehlerroutine
------------------------------------------------------------------
[BF36/E97D]
Busverzögerung für 1541 Bus gegenüber 1540 Bus
FEF3  8A         TXA                X-Register retten
FEF4  A2 05      LDX #$05           Zähler setzen
FEF6* CA         DEX                42 Taktzyklen Verzögerung
FEF7  D0 FD      BNE $FEF6          Zeit abgelaufen ?
FEF9  AA         TAX                ja, X-Register wieder herstellen
FEFA  60         RTS                zuruck zur aufrufenden Routine
------------------------------------------------------------------
```

```
[82AB/E980]
Nullbit ausgeben
FEFB  20 AE E9   JSR $E9AE       Clock Ausgang auf High setzen
FEFE  4C 9C E9   JMP $E99C       Data Ausgang auf Low setzen
---------------------------------------------------------------
[Einsprung über 'UI' Befehl]
1541/1540 Busbetrieb umschalten
FF01  AD 02 02   LDA $0202       3. Zeichen aus Befehlsstring holen
FF04  C9 2D      CMP #$2D        und auf '-' testen
FF06  F0 05      BEQ $FF0D       identisch ?
FF08  38         SEC             nein, Zeichen mit
FF09  E9 2B      SBC #$2B        '+' vergleichen
FF0B  D0 DA      BNE $FEE7       identisch ?
FF0D[1] 85 23    STA $23         ja, Flag für Busmodus setzen
FF0F  60         RTS             zurück zur aufrufenden Routine
---------------------------------------------------------------
[EAA4]
Ein-/Ausgabe initialisieren
FF10  8E 03 18   STX $1803       Datenrichtung für PA setzen
FF13  A9 02      LDA #$02        [Fehler siehe 7.1.5]
FF15  4C 5A A6   JMP $A65A       weiter
---------------------------------------------------------------
[A664]
Datenrichtung für PB setzen
FF18  A9 1A      LDA #$1A        %00011010
FF1A  8D 02 18   STA $1802       in Datenrichtungsregister
FF1D  4C A7 EA   JMP $EAA7       wieder zurück zum Reset
---------------------------------------------------------------
[E9DC/FF25]
auf Data = Low (phys. High) warten und Timer setzen
FF20  AD 00 18   LDA $1800       Bussteuerregister holen
FF23  29 01      AND #$01        und Data Leitung testen
FF25  D0 F9      BNE $FF20       ist Data gesetzt ?
FF27  A9 01      LDA #$01        nein, Zähler
FF29  8D 05 18   STA $1805       für 256 Zyklen starten
FF2C  4C DF E9   JMP $E9DF       weiter
---------------------------------------------------------------
[EE3D]
Diskette formatieren
FF2F  A9 FF      LDA #$FF        Flag für aktuelle Spur
FF31  85 51      STA $51         löschen
```

```
FF33     AD 0F 18   LDA $180F      Steuerregister holen
FF36     29 20      AND #$20       und Betriebsmodus testen
FF38     D0 03      BNE $FF3D      ist Floppy im 1541 Modus (1 MHz)
FF3A     A9 24      LDA #$24       ja, max. Zahl der Spuren festlegen
FF3C     2C         .byte $2C      nächste 2 Bytes überspringen (Bitbefehl)
FF3D[1]  A9 47      LDA #$47       Zahl der Spuren bei 2-Seiten-Betrieb
FF3F     8D AC 02   STA $02AC      Spurenzahl setzen
FF42     4C 79 A7   JMP $A779      Diskette formatieren
-----------------------------------------------------------------------
[FA32]  vgl. 87E7/9A66
einen Halbschritt nach außen
FF45     98         TYA            Y-Register
FF46     48         PHA            retten
FF47     A0 64      LDY #$64       Zahl der Abtastversuche für Spur0 (100)
FF49[1]  AD 0F 18   LDA $180F      Steuerregister A holen
FF4C     6A         ROR A          Spur0-Kennung (bit0) ins Carry
FF4D     08         PHP            und Carry merken
FF4E     AD 0F 18   LDA $180F      Steuerregister nochmal lesen
FF51     6A         ROR A          Spur0-Kennung (bit0)
FF52     6A         ROR A          nach Bit7 schieben
FF53     28         PLP            vorheriges Abtastergebnis holen
FF54     29 80      AND #$80       letztes Abtastergebnis isolieren
FF56     90 04      BCC $FF5C      ist im ersten Test Spur0 aktiv ?
FF58     10 1D      BPL $FF77      nein, ist Spur0 jetzt erreicht ?
FF5A     30 02      BMI $FF5E      ja, immer Sprung nach $FE5E
FF5C[1]  30 19      BMI $FF77      ist Spur0 immer noch aktiv ?
FF5E[1]  88         DEY            ja, noch ein Versuch
FF5F     D0 E8      BNE $FF49      schon alle Versuche durchgeführt ?
FF61     B0 14      BCS $FF77      ja, ist Kopf auf Spur0-Position ?
FF63     AD 00 1C   LDA $1C00      ja, Steuerregister für Steppermotor
FF66     29 03      AND #$03       Stepperbits isolieren
FF68     D0 0D      BNE $FF77      ist eine Stepperspule angesteuert ?
FF6A     A5 7B      LDA $7B        nein, Kopfsteuerbyte bei Lesefehlern
FF6C     D0 09      BNE $FF77      Kopf positionieren ?
FF6E     68         PLA            nein, Y-Register
FF6F     A8         TAY            wieder herstellen
FF70     A9 00      LDA #$00       Zahl der Stepperschritte
FF72     85 4A      STA $4A        löschen
FF74     4C BE FA   JMP $FABE      Kopf initialisieren
FF77[4]  68         PLA            Y-Register
FF78     A8         TAY            wieder herstellen
```

```
FF79  E6 4A     INC $4A       noch einen Schritt nach außen
FF7B  AE 00 1C  LDX $1C00     Steuerregister holen
FF7E  CA        DEX           und Kopf einen Schritt nach außen
FF7F  4C 38 FA  JMP $FA38     setzen
------------------------------------------------------------------
[903D/EBC2]
1541 Modus initialisieren
FF82  20 59 F2  JSR $F259     Diskcontroller Reset
FF85  A9 05     LDA #$05      IBM-34 Sektorversatz
FF87  85 3C     STA $3C       festlegen
FF89  A9 88     LDA #$88      IRQ Vektor auf
FF8B  8D A9 02  STA $02A9     Routine $9D88
FF8E  A9 9D     LDA #$9D      (1541 Interrupt)
FF90  8D AA 02  STA $02AA     richten
FF93  A9 24     LDA #$24      maximale Zahl der Spuren (35)
FF95  8D AC 02  STA $02AC     setzen
FF98  18        CLC           Flag für 'Seite 1'
FF99  4C F3 93  JMP $93F3     Kopf anwählen
------------------------------------------------------------------
[EE1D]
Laufwerk aktivieren
FF9C  85 FF     STA $FF       Laufwerksstatus setzen
FF9E  4C 00 C1  JMP $C100     LED anschalten
------------------------------------------------------------------
[D610]
Kopfsteuerbyte setzen
FFA1  85 7B     STA $7B       Byte in Zeiger setzen
FFA3  4C 76 D6  JMP $D676     zurück
------------------------------------------------------------------
[D628]
Positioniermodus neben Spur zurücksetzen
FFA6  20 76 D6  JSR $D676     Kopf zurücksteuern
FFA9  A9 00     LDA #$00      Flag für
FFAB  85 7B     STA $7B       Kopfmodus löschen
FFAD  60        RTS           zurück zur aufrufenden Routine
------------------------------------------------------------------
[CD91]
Pufferzeiger bei 'B-W' setzen
FFAE  A4 82     LDY $82       Kanalnummer holen
FFB0  4C DE D3  JMP $D3DE     Zeiger setzen
------------------------------------------------------------------
```

```
FFB3  FF ...                        unbenutzter
FFE5  ... FF                        ROM-Bereich
-----------------------------------------------------------------
[In 1571 DOS nicht verwendet]
DOS-Systemvektoren
FFE6  C6 C8                         Diskette formatieren        $C8C6
FFE8  8F F9                         Laufwerksmotor ausschalten  $F98F
-----------------------------------------------------------------
User-Vektoren; Sprungadressen der User-Befehle
FFEA  5F CD                         U1 oder UA  : Sektor lesen       $CD5F
FFEC  CD 97                         U2 oder UB  : Sektor schreiben   $CD97
FFFE  03 05                         U3 oder UC  : Sprung in Puffer 2 $0500
FFF0  06 05                         U4 oder UD  : Sprung in Puffer 2 $0503
FFF2  09 05                         U5 oder UE  : Sprung in Puffer 2 $0506
FFF4  0C 05                         U6 oder UF  : Sprung in Puffer 2 $0509
FFF6  05 0F                         U7 oder UG  : Sprung in Puffer 2 $050C
FFF8  0F 05                         U8 oder UH  : Sprung in Puffer 2 $050F
FFFA  01 FF                         U9 oder UI  : Umschalten 1540/41 $FF01
-----------------------------------------------------------------
Systemvektoren
FFFC  A0 EA                         U: oder UJ  : Reset durchführen  $EAA0
FFFE  67 FE                         IRQ-Vektor  (Bus/Diskkontroller) $FE67
-----------------------------------------------------------------
```

7.2.3 Das C-1570-ROM im Vergleich

Die C-1570 hat fast das gleiche Betriebssystem wie die C-1571. Es handelt sich dabei um ein Modifizertes 1571-ROM. Besonders einige technischen Komponenten sind bei der C-1570 anders als bei der C-1571.

Der gravierenste Unterschied ist natürlich, daß die C-1570 keine zweiten Schreib-/Lesekopf hat. Aus diesem Grund wurden alle Stellen im ROM, die auf die zweite Diskettenseite umschalten geändert. Auch die Verwaltung der BAM geschieht nun wie bei der VC-1541.

Desweiteren unterscheidet sich die C-1570 in ihren beiden Motoren. Sowohl der Laufwerks- als auch der Steppermotor sind nicht so leistungsfähig, wie bei der C-1571. Die Motoren der C-1570 sind nicht so schnell auf Drehzahl oder benötigen langsamere Stepperimpulse. Deshalb wurden die für die Motorsteuerung verantwortlichen Zeitkonstanten geändert.

Zum Schluß wurde noch einige kleine Fehler des C-1570 ROMs korregiert. Dabei ist bemerkenswert, daß die Seriengeräte der C-1571 noch diese Fehler enthalten und trotzdem funktionieren nd verkauft werden.

C-1570 DOS

```
--------------------------------------------------------------------
8000  75 98                     Prüfsumme des 1570 ROM
--------------------------------------------------------------------
84E4  20 4D AA   JSR $AA4D      Diskette formatieren
--------------------------------------------------------------------
8827  A0 08      LDY #$08       Verzögerung für 1541 Steppermotor
--------------------------------------------------------------------
8FD4  4C 21 90   JMP $9021      '31 Syntax Error' ausgeben
--------------------------------------------------------------------
90D9  20 5B AA   JSR $AA5B      Dateityp auf 'PRG' prüfen
--------------------------------------------------------------------
A40F  8D D7 FE   STA $FED7      höchste Spurnummer setzen
```

```
----------------------------------------------------------------------
A445  60          RTS           Teil für 2. Diskettenseite
A446  EA          NOP           abtrennen
----------------------------------------------------------------------
A4DD  53 54 45 56 45 20 4C 41   Copyright Vermerk für 1570 Modifikation
A4E5  4D 0D                     'STEVE LAM'
----------------------------------------------------------------------
A597  AD D7 FE    LDA $FED7     Nummer der höchsten Spur holen
----------------------------------------------------------------------
A671  AD D7 FE    LDA $FED7     Nummer der höchsten Spur holen
----------------------------------------------------------------------
A6C4  8D D7 FE    STA $FED7     höchste Spurnummer setzen
----------------------------------------------------------------------
A6DF  8D D7 FE    STA $FED7     höchste Spurnummer setzen
----------------------------------------------------------------------
A6F6  8D D7 FE    STA $FED7     höchste Spurnummer setzen
----------------------------------------------------------------------
A726  8D D7 FE    STA $FED7     höchste Spurnummer setzen
----------------------------------------------------------------------
A773  8D D7 FE    STA $FED7     höchste Spurnummer setzen
----------------------------------------------------------------------
A7B3  20 62 AA    JSR $AA62     Steuerregister holen
----------------------------------------------------------------------
A7C7  4C CE A7    JMP $A7CE     1541 Modus nicht testen
----------------------------------------------------------------------
A7D7  AD D7 FE    LDA $FED7     Nummer der höchsten Spur holen
----------------------------------------------------------------------
A941  AD D7 FE    LDA $FED7     Nummer der höchsten Spur holen
----------------------------------------------------------------------
A597  AD D7 FE    LDA $FED7     Nummer der höchsten Spur holen
----------------------------------------------------------------------
[D867]  Fehlerrückmeldung prüfen
AA3F  C9 02       CMP #$02      mit erster Fehlernummer vergleichen
AA41  90 07       BCC $AA4A     liegt Fehler vor?
AA43  C9 0F       CMP #$0F      nein, auf 'Drive Not Ready' prüfen
AA45  F0 03       BEQ $AA4A     ist Laufwerk bereit?
AA47  4C 6B D3    JMP $D36B     ja, zurück zur Hauptroutine
AA4A  4C 73 D3    JMP $D373     Fehler beachten
```

```
-------------------------------------------------------------------------
[84E4]  Diskette formatieren
AA4D  85 51      STA $51        aktuelle zu formatierende Spur setzen
AA4F  20 7C 87   JSR $877C      LED am Laufwerk anschalten
AA52  20 89 A9   JSR $A989      Diskette formatieren
AA55  48         PHA            Rückmeldung retten
AA56  20 88 87   JSR $8788      LED am Laufwerk ausschalten
AA59  68         PLA            Rückmeldung wieder holen
AA5A  60         RTS            zurück zur aufrufenden Routine
-------------------------------------------------------------------------
[90D9]  Dateityp auf Programmdatei testen
AA5B  A5 E7      LDA $E7        Dateityp-Byte holen
AA5D  29 07      AND #$07       und Flags für Dateityp isolieren
AA5F  C9 02      CMP #$02       mit 'PRG' vergleichen
AA61  60         RTS            zurück zur aufrufenden Routine
-------------------------------------------------------------------------
[AA62]  Steuerregister lesen
AA62  AD 0F 18   LDA #$180F     Wert des Steuerregisters holen
AA65  2C 01 18   BIT #$1801     neuen Wert in Steuerregister einlesen
AA68  60         RTS            zurück zur aufrufenden Routine
-------------------------------------------------------------------------
CD22  CD D7 FE   CMP $FED7      mit maximaler Spur vergleichen
-------------------------------------------------------------------------
D03F  4C 8C D5   JMP $D58C      Job ausführen
-------------------------------------------------------------------------
D05D  20 86 D5   JSR $D586      BAM von Diskette lesen
-------------------------------------------------------------------------
D097  9D FA 02   STA $02FA,X    High-Byte der Zahl der freien Blocks
-------------------------------------------------------------------------
D367  4C 3F AA   JMP $AA3F      Rückmeldung prüfen
-------------------------------------------------------------------------
D51D  CD D7 FE   CMP $FED7      mit maximaler Spur vergleichen
-------------------------------------------------------------------------
D563  CD D7 FE   CMP $FED7      mit maximaler Spur vergleichen
-------------------------------------------------------------------------
E5C7  B0                        letztes Zeichen der Einschaltmeldung
-------------------------------------------------------------------------
ED8F  20 B7 EE   JSR $EEB7      neue BAM erzeugen
-------------------------------------------------------------------------
EE40  20 05 F0   JSR $F005      Puffer für BAM setzen
```

```
--------------------------------------------------------------------------
EEB1  20 60 D4   JSR $D460        Sektor lesen
--------------------------------------------------------------------------
EF28  20 20 F2   JSR $F220        Zahl der freien Blöcke überprüfen
--------------------------------------------------------------------------
EF2F  CD D7 FE   CMP $FED7        mit maximaler Spur vergleichen
--------------------------------------------------------------------------
EF37  4C 8A D5   JMP $D58A        BAM auf Diskette schreiben
--------------------------------------------------------------------------
EF5F  20 CF EF   JSR $EFCF        BAM-Pufferzeiger setzen
--------------------------------------------------------------------------
EF93  20 CF EF   JSR $EFCF        BAM-Pufferzeiger setzen
--------------------------------------------------------------------------
F001  4C 8A D5   JMP $D58A        BAM auf Diskette schreiben
--------------------------------------------------------------------------
F005  20 3A EF   JSR $EF3A        Pufferzeiger setzen
--------------------------------------------------------------------------
F09C  4C 8A D5   JMP $D58A        BAM auf Diskette schreiben
--------------------------------------------------------------------------
F107  4C 86 D5   JMP $D586        BAM von Diskette lesen
--------------------------------------------------------------------------
F12D  A5 6F      LDA $6F          aktuellen BAM-Zeiger
F12F  48         PHA              retten
--------------------------------------------------------------------------
F147  CD D7 FE   CMP $FED7        mit maximaler Spur vergleichen
--------------------------------------------------------------------------
F1C4  20 11 F0   JSR $F011        BAM-Zeiger setzen
--------------------------------------------------------------------------
F1D5  CD D7 FE   CMP $FED7        mit maximaler Spur vergleichen
--------------------------------------------------------------------------
F1DF  20 11 F0   JSR $F011        BAM-Zeiger setzen
--------------------------------------------------------------------------
F24B  AE D6 FE   LDX $FED6        Zahl der Spurzonen der Diskette
--------------------------------------------------------------------------
F98A  A9 7D      LDA #$7D         Verzögerung bei 'Motor an' auf 1/0.5 sec
--------------------------------------------------------------------------
FF3F  8D D7 FE   STA $FED7        maximale Spurnummer merken
--------------------------------------------------------------------------
FF95  8D D7 FE   STA $FED7        maximale Spurnummer merken
--------------------------------------------------------------------------
```

KAPITEL 8

DIE FLOPPY IM GRIFF - ÜBERSICHTSTABELLEN

8.1 1571 ZEROPAGE LISTING

```
.........................................................................
  0 -    5 Jobcode des Auftrags für entsprechenden Puffer (0-5)
 $0 -   $5 Puffer 5 ist im RAM nicht angelegt.

           Bedeutung der Jobcodes»:

           $80 Lesen eines Sektors
           $88 Sektor auf gleicher Spur lesen
           $90 Schreiben eines Sektors
           $A0 Verify eines Sektors
           $B0 Suchen eines Sektorheaders
           $C0 Kopf auf Spur0 setzen
           $D0 Programm im Puffer ausführen
           $E0 Programm in Jobschleife einbinden
           $F0 Diskette formatieren

           Bedeutung der Rückmeldungen»:

           $00/$01 kein Fehler aufgetreten
           $02 Blockheader nicht gefunden
           $03 Sync-Markierung nicht gefunden
           $04 Datenblock nicht gefunden
           $05 Datenblockprüfsumme fehlerhaft
           $06 Formatierfehler
           $07 Verify Fehler
           $08 Schreibschutz vorhanden
           $09 Headerprüfsumme fehlerhaft
           $0A Datenblock zu lang
           $0B falsche ID / Diskette gewechselt
           $0D Indexloch nicht gefunden
           $0E Syntaxfehler bei CP/M Kommandos
           $0F keine Diskette eingelegt

.........................................................................
  6 -   17 jeweils Spur. und Sektornummer für Puffer 0 bis 5
 $6 -  $11 So ist z.B in 6 die Spur für Puffer 0 und in 7 die
           dazugehörige Sektornummer
```

18 - 19		erstes und zweites Zeichen der ID der Diskette
$12 - $13		in Laufwerk 0
20 - 21		unbenützte
$14 - $15		Speicherstellen
22 - 23		erstes und zweites Zeichen der ID des zuletzt
$16 - $17		gelesenen Sektorheaders
24 - 25		Spur. und Sektornummer des zuletzt
$18 - $19		gelesenen Sektorheaders
26		Prüfsumme des zuletzt gelesenen
$1A		Sektorheaders
27		Steuerbyte der Routine $86E6
$1B		Pufferzeiger in Formatierroutine $9B89
28		Flag für 'Diskette initialisieren'
$1C		0= nein <>= ja
29		Wie 28/$1C, nur für Laufwerk 1
$1D		Wert immer 1 [EBBA]
30		aktueller Zustand der Schreibschutzkerbe
$1E		0= Schreibschutz aktiv 1=kein Schreibschutz angebracht
31	$1F	unbenützt
32		Betriebsstatus des Laufwerk 0
$20		Bit4 : 1= Motor läuft nach, soll ausgeschaltet werden
		Bit5 : 1= Motor ist angeschaltet
		Bit6 : 1= Steppermotor ist aktiv, Kopf wird gesetzt
		Bit7 : 1= Laufwerk ist noch nicht bereit
33	$21	Wie 32/$20, nur für Laufwerk 1
34	$22	Spurnummer des aktuellen Jobs
35	$23	Flag für Busmodus : 0= 1541 Bus <>0= 1540 Bus

....

```
 36 -   43 Puffer für Sektorheader
$24 -  $2B Commodore Sektoren : Daten in GCR-Code
           CP/M Sektoren      : ID Feldinhalt
```

....

```
 44 -   45 unbenützte
$2C -  $2D Speicherstellen
```

....

```
 46 -   47 Zeiger auf aktuelle Pufferposition
$2E -  $2F bei GCR Umwandlung
```

....

```
 48 -   49 Zeiger auf Position in
$30 -  $31 aktuellen Datenpuffer
```

....

```
 50 -   51 Zeiger auf Spur-/Sektortabelle der Jobschleife (6-17)
$32 -  $33 beim Formatieren: Zeiger auf aktuellen Sektorheader
```

....

```
 52    $34 Zeiger auf aktuelles GCR-Byte bei Umwandlung
```

....

```
 53    $35 noch auszuführende Jobschleifen bei 'Motor aus'
```

....

```
 54    $36 Zeiger auf Position des Binärbytes bei GCR-Umwandlung
```

....

```
 55    $37 Busstatusbyte :
           Bit0 1=Flag für 'Datei hat nur einen Sektor'
           Bit3 umgekehrter Zustand der Clock-Leitung
                nächstes, zu erwartendes Clock-Signal
           Bit6 1= 1571 Busmodus   0= 1541 Busmodus
           Bit7 1= 1571 Betriebsmodus (2 MHz Takt)
                0= 1541 Betriebsmodus (1 MHz Takt)
```

....

```
 56    $38 Kennzeichen des zuletzt gelesenen Headers (normal 7)
```

....

```
 57    $39 Kennzeichen für Datenblock (8)
```

....

```
 58    $3A Prüfsumme des Puffers / Datenblocks
```

....

```
 59    $3B Befehlsnummer bei 'U0' [siehe $8030]
```

....

```
 60    $3C Sektorversatz bei IBM-System-34 (nach Reset 5)
```

61	$3D	Laufwerksnummer des aktuellen Jobs
62	$3E	Nummer des gerade aktiven Laufwerks ($FF= kein Laufwerk)
63	$3F	Puffernummer des aktuellen Jobs
64	$40	Spur des letzen Jobs
65	$41	Nummer des letzten Jobs [nur verwendet in F340/F44B]
66	$42	Differenz zwischen neuer und alter Spur
67	$43	Zahl der Sektoren der Spur
68	$44	Zahl der CP/M Sektorabschnitte pro Sektor
69	$45	Befehlbits des Jobcodes (Bit 3-6 des Orginaljobcodes)
70	$46	nächstes auf 1571 Bus auszugebendes Zeichen
71	$47	Kennzeichen für Datenblock (8)
72	$48	Zähler für Nachlaufzeit des Motors
73	$49	Zwischenspeicher für Stackpointer
74	$4A	Bit0-6 : Zahl der zu fahrenden Halbspurschritte Bit7 : 1= Schritte nach außen 0= Schritte nach innen
75	$4B	diverser Zwischenspeicher
76	$4C	Sektordifferenz zu nächstem optimalem Job
77	$4D	Sektornummer des nächsten optimalen Jobs
78 - 79 $4E - $4F		Zwischenspeicher für aktuellen Pufferzeiger bei GCR Umwandlung

80	$50	Flag für Pufferdatenformat 0= Binärdaten <>0= GCR-Daten
81	$51	aktuelle Spur beim Formatieren $FF= Formatieren nicht im Gang
82 - 85 $52 - $55		Zwischenspeicher für 4 Binärbytes, die in 5 GCR-Bytes umgewandelt werden sollen
86 - 93 $56 - $5D		Zwischenspeicher für 8 GCR-Werte, die 8 Binärhalbbytes und somit 4 Binärbytes ergeben
94	$5E	Befehlsstatusbyte Bit 0-4 : letzte Fehlermeldung der Jobschleife unter CP/M Bit7 : 1= Diskette im IBM-System-34 Format
95	$5F	aktueller Jobcode
96	$60	IBM-34-Format : kleinste Sektornummer der Spur
97	$61	IBM-34-Format : größte Sektornummer der Spur
98 - 99 $62 - $63		1541 Modus : Zeiger auf aktuelle Kopfsteuerroutine 1571 Modus : Zeiger auf Positionierphase
100	$64	aktuelle Position des Kopfs in Halbspurschritten
101 - 102 $65 - $66		Zeiger auf Reset (kein Hardwaretest) $EB22 wird von $FEE7 angesprungen, wenn 'UI' nicht '+' oder '-' hat
103	$67	anzusteuernde Zielspur
104	$68	Flag für Initialisiermethode (immer 0) [gesetzt durch C63D] 0 = Disketten automatisch initialisieren <>0 = Disketten nur von Hand initialisieren (i-Befehl)
105	$69	Sektorversatz für Commodore Disketten (6)

```
.....................................................................
 106     $6A Bit0-5 : Zahl der Leseversuche bei Fehlern
             Bit6   : 1= Kopf nicht neben Spur setzen
             Bit7   : 1= Spur0 nicht anfahren
.....................................................................
 107 -   108 Zeiger auf Tabelle der 1541 User Befehle
 $6B -   $6C (Adresse $FFEA)
.....................................................................
 109 -   110 Zeiger in BAM-Puffer
 $6D -   $6E
.....................................................................
 111 -   114 Zwischenspeicher für diverse Zwecke
 $6f -   $72 (BAM-Berechnungen, etc.)
.....................................................................
 115     $73 Zahl der Side-Sektoren der relativen Datei
.....................................................................
 116     $74 unbenützte Speicherstelle
.....................................................................
 117 -   118 Adressenzeiger für verschiedene
 $75 -   $76 Systemzwecke
.....................................................................
 119     $77 Geräteadresse für Listen + $20 (Flag in Kommandobyte)
.....................................................................
 120     $78 Geräteadresse für Talk + $40 (Flag in Kommandobyte)
.....................................................................
 121     $79 Flag für Listen (1= Listener Betrieb)
.....................................................................
 122     $7A Flag für Talk (1= Talker Betrieb)
.....................................................................
 123     $7B aktuelles Kopfsteuerbyte für Positionierung bei Lesefehlern
.....................................................................
 124     $7C Flag für 'ATN aufgetreten'
.....................................................................
 125     $7D Flag für 'ATN beachten'  0= ja;  <>0= ATN ignorieren
.....................................................................
 126     $7E Spurnummer des letzten Zufgriffs
.....................................................................
 127     $7F Nummer des aktuellen Laufwerks
.....................................................................
 128     $80 Nummer der aktuellen Spur
```

129 $81 aktuelle Sektornummer

130 $82 Nummer des aktuellen internen Kanals (0-6)

131 $83 aktuelle Sekundäradresse

132 $84 letztes auf IEC-Bus gesendetes Kommandowort

133 $85 aktuelles Datenbyte für 1541 Bus

134 - 138 Zwischenspeicher für
$86 - $8A diverse Zwecke

139 - 142 Rechenregister 1
$8B - $8E

143 - 147 Rechenregister 2
$90 - $94

148 - 149 Zeiger in Directorypuffer
$94 - $95

150 $96 Nummer des zuerst gelesenen IBM-34 Sektors

151 $97 Zahl der IBM-34 Sektoren pro Spur

152 $98 Bitzähler für Bits pro Byte (bei Datenübertragung)

153 - 154 Zeiger auf Startadresse von
$99 - $9A Puffer 0 ($0300)

155 - 156 Zeiger auf Startadresse von
$9B - $9C Puffer 1 ($0400)

157 - 158 Zeiger auf Startadresse von
$9D - $9E Puffer 2 ($0500)

159 - 160 Zeiger auf Startadresse von
$9F - $A0 Puffer 3 ($0600)

```
.....................................................................
 161 -  162 Zeiger auf Startadresse von
 $A1 -  $A2 Puffer 4 ($0700)
.....................................................................
 163 -  164 Zeiger auf Startadresse des
 $A3 -  $A4 Eingabepuffers ($0200)
.....................................................................
 165 -  166 Zeiger auf Startadresse des
 $A5 -  $A6 Fehlermeldungspuffers ($02D5)
.....................................................................
 167 -  173 Kanal-Puffer-Tabelle 1 :
 $A7 -  $AD Zuordnung eines ersten Puffers zu internem Kanal
            167-173 entspricht Kanal 0-6

            Bedeutung des Bytes :

            Bit0-5   :  Puffernummer die Kanal zugeordnet ist
            Bit6     :  1= Pufferinhalt zurückschreiben
            Bit7     :  0= Puffer gerade aktiv benützt
            Wert $FF :  kein Puffer zugeteilt
.....................................................................
 174 -  180 Kanal-Puffer-Tabelle 2 :
 $AE -  $B4 ordnet 2. Puffer zu (Funktion wie 167-173)
.....................................................................
 181 -  186 Zahl der von der über internen Kanal angesprochenen Datei
 $B5 -  $BA belegten Blocks (Low-Byte)
            Index : Kanalnummer $82
.....................................................................
 187 -  192 Zahl der von der über internen Kanal angesprochenen Datei
 $BB -  $C0 belegten Blocks (High-Byte)
            Index : Kanalnummer $82
.....................................................................
 193 -  198 Zeiger auf aktuelles Datenbyte der Datei
 $C1 -  $C6 über internen Kanal
            Index : Kanalnummer $82
.....................................................................
 199 -  204 Recordlänge der über internen Kanal
 $C7 -  $CC geöffneten relativen Datei
            Index : Kanalnummer $82
```

```
.................................................................
 205 -  210 Kanal-Puffer-Tabelle 3 :
 $CD -  $D2 ordnet 3. Puffer zu (Funktion wie 167-173)
.................................................................
 211    $D3 Zeiger auf ersten Dateinamen
.................................................................
 212    $D4 Position in aktuellem Record
.................................................................
 213    $D5 Side-Sektor-Nummer
.................................................................
 214    $D6 Zeiger auf Record in Side-Sektor
.................................................................
 215    $D7 Zeiger auf Datensatz bei relativer Datei
.................................................................
 216 -  220 Tabelle Dateiname-Diretorysektor
 $D8 -  $DC Directorysektor, wo Dateiname vorkommt
.................................................................
 221 -  225 Tabelle Dateiname-Position in Directorysektor
 $DD -  $E1 merkt Stelle des Directoryeintrags
.................................................................
 226 -  230 Tabelle Dateiname-angesprochenes Laufwerk
 $E2 -  $E6
.................................................................
 231 -  235 Tabelle Dateiname-Dateityp
 $E7 -  $EB
.................................................................
 236 -  241 Tabelle Kanalnummer-Dateityp
 $EC -  $F1 Bit0   :  Laufwerksnumer (0/1)
            Bit1-3:  Dateityp
.................................................................
 242 -  247 Tabelle Kanalnummer-Status
 $F2 -  $F7 Bit1   :  1= Kanal ist Schreibkanal
            Bit3   :  0= EOI Flag gesetzt
            Bit7   :  1= Kanal ist Lesekanal
.................................................................
 248    $F8 Flag für EOI (letztes Zeichen);  0= ja  1= nein
.................................................................
 249    $F9 Nummer des aktuellen Puffers
.................................................................
 250 -  254 Tabelle Puffer-interne Kanalnummer
 $FA -  $FE
```

```
.....................................................................
 255    $FF Laufwerksstatus für Laufwerk 0;  0= Laufwerk bereit
.....................................................................
 256   $100 Laufwerksstatus für Laufwerk 1;  0= Laufwerk bereit
.....................................................................
 257 -  325 Hardwarestack des
$101 - $145 Prozessors
.....................................................................
 326 -  431 BAM-Puffer 2 für 1571 Disketten
$146 - $1AF
.....................................................................
 432   $1B0 Fehlermeldung unter CP/M Betrieb
.....................................................................
 433   $1B1 Flag für aktuelle Diskettenseite;  0= Seite 1
.....................................................................
 443 -  511 Zusatzpuffer um GCR-Daten aufzunehmen
$1BB - $1FF
.....................................................................
 512 -  553 Eingabepuffer für Befehlsstrings
$200 - $229 vom Rechner
.....................................................................
 554   $22A Nummer des aktuellen Befehls;  $FF= kein Befehl vorhanden
.....................................................................
 555 -  573 Tabelle Sekundäradresse-interner Kanal
$22B - $23D Bit0-3  :  interne Kanalnummer
            Bit6    :  1= Kanal zum Lesen (Talker)
            Bit7    :  1= Kanal zum Schreiben (Listener)
            Wert $FF:  Sekundärkanal nicht belegt
.....................................................................
 574 -  579 Tabelle Kanalnummer-aktuelles Datenbyte
$23E - $243
.....................................................................
 580 -  585 Tabelle Kanalnummer-Zahl der noch zu übertragenden Bytes
$244 - $249
.....................................................................
 586   $24A aktueller Dateityp
.....................................................................
 587   $24B Länge des aktuellen Dateinamens im Befehlsstring
.....................................................................
 588   $24C Zwischenspeicher für Sekundäradresse bei Open
```

589 $24D bei Aufruf von D506 in Jobcode einblenden

590 $24E maximale Zahl der Sektoren der aktuellen Spur

591 - 592 Tabelle der Pufferbelegung
$24F - $250 jedes Bit des 16-Bit-Wert repräsentiert einen Puffer
1= Puffer belegt; 0= Puffer frei

593 $251 Flag für 'BAM neu schreiben, da ungültig'; 1=ja 0=nein

594 $252 Wie oben nur für Laufwerk 1

595 $253 Flag für 'Dateieintrag gefunden'; $FF= nein

596 $254 Flag für 'Directory im Puffer'; 0=ja <>0=nein

597 $255 Befehlsmodus

598 $256 Bitmap für Kanalbelegung
1= Kanal frei; 0=Kanal belegt

599 $257 Zeiger auf aktuell aktiven Puffer bei 2-Pufferbetrieb

600 $258 Recordlänge

601 $259 aktueller Side-Sektor (Spur)

602 $25A aktueller Side-Sektor

603 - 607 Tabelle Puffer-Jobcode
$25B - $25F letzer Jobcode des Puffers

608 - 613 Tabelle Kanal-Datensektor (Spurnummer)
$260 - $265

614 - 619 Tabelle Kanal-Datensektor (Sektornummer)
$266 - $26B

620 $26C Fehlernummer / Blinkzähler

....

621 $26D LED-Maske bei Fehlerblinken

....

622 $26E letztes aktives Laufwerk [D7D1/D9FE]

....

623 $26F Nummer des letzen Sektors [D7DC/DA03]

....

624 $270 aktuelle Kanalnummer

....

625 $271 Zahl der Bytes pro IBM-34 Sektorabschnitt

....

626 - 627 Zwischenspeicher bei Directoryerzeugung
$272 - $273 (z.B. für Blockzahl, etc.)

....

628 $274 Länge des Befehlsstrings im Eingabepuffer

....

629 $275 in Eingabepuffer zu suchendes Zeichen [C165/C16D/C268/C273]

....

630 $276 Länge des aktuellen Dateinamens im Eingabepuffer

....

631 $277 Zahl der Dateinamen für 1. Dateinennung

....

634 - 639 Tabelle Dateiname-Position im Eingabepuffer
$27A - $27F zeigt auf Beginn in Befehlsstring

....

640 - 644 Tabelle Dateiname-Spur des aktuellen Sektors
$280 - $284

....

645 - 649 Tabelle Dateiname-Nummer des aktuellen Sektors
$285 - $289

....

650 $28A Flag für Joker; 0= kein Joker

....

651 $28B Befehlssyntaxbyte

....

652 $28C Zahl der zuzugreifenden Laufwerke (0/1/2)

....

653 $28D Flag für Directory von beiden Laufwerken; 0= nein

....

654 $28E Nummer des letzten Laufwerks

655	$28F	Position des aktuellen Directoryeintrags
656	$290	Sektor des aktuellen Dateieintrags
657	$291	Sektor des aktuellen Dateieintrags
658	$292	Zeiger auf gültigen Eintrag
659	$293	Zeiger auf nächsten Directorysektor
660	$294	Position im Directorysektor
661	$295	Zähler für Directoryeinträge pro Sektor (8)
662	$296	Dateityp aus Befehlsstring; 0= keine Angabe
663	$297	Dateibetriebsmodus 0/1= Lesen/Schreiben 2=Append 3=Modify
664	$298	Flag für 'Fehler bei Job beachten'; >128= nein <128= ja
665	$299	Zeiger auf Positionierphase bei Lesefehlern
666	$29A	Steuerbyte für Kopfpositionierung bei Lesefehlern
667 - 668	$29B - $29C	Zeiger auf aktuellen BAM-Spur Zwischenspeicher für Laufwerk 0 und 1
669 - 670	$29D - $2A0	Nummer der Spuren durch die die nachfolgenden BAM-Zwischenspeicher belegt sind
673 - 680	$2A1 - $2A8	BAM Zwischenspeicher
681 - 682	$2A9 - $2AA	IRQ - Vektor aus FE67
683	$2AB	Zähler für Laufzeit des Motors beim Disketten einlegen
684	$2AC	Nummer der größten Spur +1 der Diskette

....

685 - 686 Zeiger in BAM-Puffer
$2AD - $2AE (Zwischenspeicher um Zeiger zu retten)

....

687 $2AF Flag für 'von 1541 auf 1571 IRQ umschalten'; 1= nein

....

688 - 715 Puffer um Directoryzeile zu
$2B0 - $2CB erzeugen

....

716 - 724 unbenützter Bereich
$2CC - $2D4

....

725 - 760 Puffer um Fehlerklartextmeldung zu
$2D5 - $2F8 generieren

....

761 $2F9 Flag für 'BAM ungültig'; 0=nein $C0=ja

....

762 - 763 Zahl der freien Blocks der Diskette in
$2FA - $2FB Laufwerk 0 und 1 (Low-Bytes)

....

764 - 765 Zahl der freien Blocks der Diskette in
$2FC - $2FD Laufwerk 0 und 1 (High-Bytes)

....

766 - 767 Steuerbyte für Positionierung neben Spur
$2FE - $2FF für Laufwerk 0 und 1

....

8.2 FLOPPY-ERRORS IM ÜBERBLICK

Nummer	Fehlerbedeutung
00	OK-Meldung
01	FILES SCRATCHED,XX Rückmeldung nach dem Löschen XX gibt die Zahl der gelöschten Dateien an

Die Angaben TT und SS sind die Spur- und Sektornummer des Datenblocks, wo der Fehler auftrat.

20	READ ERROR,TT,SS Der Sektorheader eines Blocks wurde nicht gefunden. Dabei handelt es sich um eine unformatierte oder beschädigte Diskette.
21	READ ERROR,TT,SS Die Sync-Markierung wurde nicht gefunden. Entweder ist die Floppy nicht formatiert oder es liegt ein Fehler am Laufwerk, beispielsweise ein dejustierter Lesekopf, vor.
22	READ ERROR,TT,SS Der Datenblock eines Sektors wurde nicht gefunden.
23	READ ERROR,TT,SS Es wurde ein Prüfsummenfehler festgestellt. In diesem Fall müssen Sie versuchen, den Sektor noch zu retten, indem Sie mit den Direktzugriffsbefehlen den Sektor mehrmals lesen, bis der Fehler eventuell nicht mehr auftritt. Ansonsten müssen Sie den Sektor in den Floppy-Puffer einlesen und neu zurückschreiben. Dabei wird die Prüfsumme neu berechnet, allerdings kann der Inhalt des Sektors fehlerhaft sein

25 WRITE ERROR,TT,SS
Beim Schreiben eines Sektors wurde beim nachträglichen Verify ein Fehler festgestellt. Sie sollten in diesem Fall eine neue Diskette verwenden.

26 WRITE PROTECT ON,TT,SS
Die Diskette ist durch eine Schreibschutzmarke geschützt.

27 READ ERROR,TT,SS Im Sektorheader wurde ein Prüfsummenfehler festgestellt.

29 DISK ID MISMATCH,TT,SS
Die ID des Sektorheaders stimmt nicht mit der zuletzt gelesenen ID überein. Maßnamen: Diskette initialisieren oder neu formatieren.

----

30 SYNTAX ERROR
Die C-1570/71 kennt den Befehl, der über den Befehlskanal gesendet wurde nicht.

31 SYNTAX ERROR
Der Befehl kann von der C-1570/71 nicht ausgeführt werden. (z.B. Backup)

32 SYNTAX ERROR
Der über den Befehlskanal gesendete Befehl ist länger als 41 Zeichen und der Eingabepuffer der Floppy ist gefüllt.

33 SYNTAX ERROR
Der Joker wurde beim Schreiben als Dateinamen verwendet.

34 SYNTAX ERROR
Der Dateiname wurde nicht gefunden. Sie haben eventuell den Doppelpunkt nach dem Befehlsbuchstaben vergessen.

39 FILE NOT FOUND
Die angegebene Autostart-Datei wurde auf der Diskette nicht gefunden.

50 RECORD NOT PRESENT
Der angesprochene Datensatz einer relativen Datei ist nicht angelegt. Beim ersten Schreiben des Datensatzes kann diese Meldung ignoriert werden. Beim Lesen weißt sie darauf hin, daß der gewünschte Datensatz nicht existiert.

51 OVERFLOW IN RECORD
Die an die Floppu übertragenen Daten sind länger als der Datensatz, wobei die überzähligen Zeichen ignoriert werden.

52 FILE TOO LARGE
Die Nummer des letzen Datensatzes ist zu groß, da eine derartige Datei nicht mehr auf die Diskette passen würde.

60 WRITE FILE OPEN
Es wurde versucht auf eine nicht ordnungsgemäß geschlossene Datei zuzugreifen. Diese kann nur im 'Modify'-Modus eröffnet werden.

61 FILE NOT OPEN
Es wurde versucht eine Datei zu benützten, die nicht geöffnet wurde.

62 FILE NOT FOUND
Das angegebene Programm oder die Datei wurden nicht gefunden.

63 FILE EXISTS
Die neue Datei ist auf der Diskette bereits vorhanden

64 FILE TYPE MISMATCH
Der beim Eröffnen angegebene Dateityp stimmt nicht mit dem im Directory verzeichneten Dateityp überein.

65 NO BLOCK,TT,SS
Der bei Block-Allocate angegeben Block ist bereits belegt. TT und SS geben die Spur- und Sektornummer des nächsten freien Blocks der Spur an. Haben TT und SS den Wert 0, dann ist kein freier Sektor mehr vorhanden. Bitte beachten Sie das in Kapitel 2.1.3 angegebene Fehlverhalten des Block-Allocate-Befehls.

66 ILLEGAL TRACK OR SECTOR,TT,SS
Die beim Direktzugriffsbefehl angegebenen Sektorparameter sind fehlerhaft.

67 ILLEGAL TRACK OR SECTOR,TT,SS
Die Sektorverkettung zeigt auf einen Sektor, der nicht vorhanden ist.

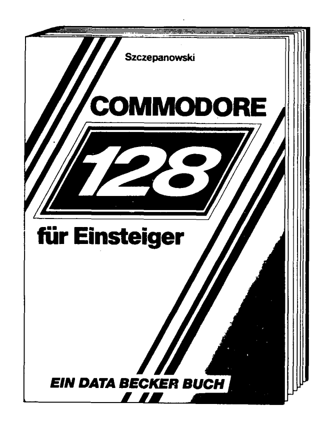
70 NO CHANNEL
Es ist kein weiterer Kanal mehr verfügbar. Sie müssen zuerst eine noch offene Datei wieder schließen, damit wieder ein Kanal zur Verfügung steht.

71 DIR ERROR,TT,SS
Das BAM-Verzeichnis im Floppy-Speicher stimmt nicht mit dem BAM-Verzeichnis auf der Diskette überein. Sie müssen in diesem Fall die Diskette initialisieren

72 DISK FULL
Die Kapazität der Diskette ist erschöpft und es sind weniger als drei Blocks frei.

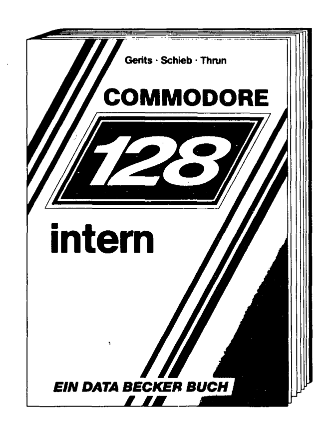
73 Einschaltmeldung
Es wurde versucht eine Diskette zu beschreiben, die unter einem anderen DOS formatiert wurde.

74 DRIVE NOT READY
Es ist keine formatierte Diskette eingelegt.



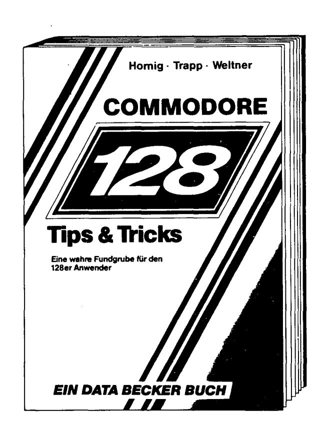
Sie wollen mit dem Commodore 128 in die Computerwelt einsteigen? Dann brauchen Sie dieses Buch! Behandelt werden: Peripheriegeräte, Tastaturbedienung, Laden und Starten von Fertigprogrammen, BASIC und Erstellung eines Adressenverwaltungsprogramms. Handbücher sind oft zu knapp und trocken geschrieben. Dieses Buch nicht!

Szczepanowski
Commodore 128 für Einsteiger
ca. 250 Seiten, DM 29,–
Erscheint im November
ISBN 3-89011-099-1



Ein Muß für jeden, der sich intensiver mit dem C-128 beschäftigt. Einführung in das System, Hardware- und Interfacebeschreibung, Erläuterung des VIC-Chips, des VDC, SID, detailliert und leichtverständliche Beschreibung der Memory-Management-Unit (MMU), ein sehr ausführlich kommentiertes ROM-Listing, Einführung: wie arbeite ich mit ROM-Listing und Zeropage, mit sehr vielen Programmbeispielen!

Gerits/Schieb/Thrun
128 INTERN
507 Seiten, DM 69,–
ISBN 3-89011-098-3



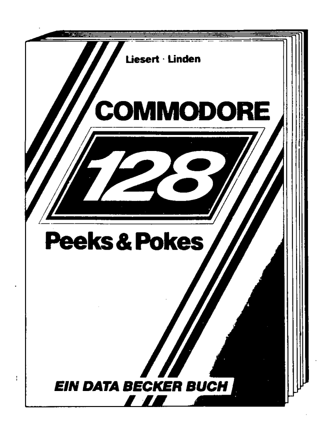
Eine Fundgrube für alle C-128 Besitzer! Ob man einen eigenen Zeichensatz erstellen, die doppelte Rechengeschwindigkeit im 64er Modus benutzen oder die vorhandenen ROM-Routinen verwenden will. Dieses Buch ist randvoll mit wichtigen Informationen; z.B.: Bank-Switching/ Speicherkonfiguration, Registererläuterungen zum Video-Controller und 640 x 200 Punkte Auflösung. Dieses Buch darf bei keinem 128er fehlen!

Hornig/Weltner/Trapp
128 TIPS & TRICKS
327 Seiten, DM 49,–
ISBN 3-89011-097-5



Sie haben den Einstieg auf dem Commodore 128 geschafft? Dann werden Sie mit diesem Buch zum Profi. Aus dem Inhalt: Datenfluß- und Programmablaufpläne, fortgeschrittene Programmiertechniken, Menueerstellung, Grafikprogrammierung, mehrdimensionale Felder, Sortierroutinen, Dateiverwaltung und viele nützliche Utilities. So lernen Sie professionelles Programmieren.

Kampow
Das große BASIC-Buch zum COMMODORE C-128
ca. 450 Seiten, DM 39,–
ISBN 3-89011-114-9



Schlagen Sie dem Betriebssystem Ihres C-128 ein Schnippchen. Wie? Mit PEEKS & POKES natürlich! Dieses Buch erklärt Ihnen leichtverständlich den Umgang damit. Mit einer riesigen Anzahl wichtiger POKES und ihren Anwendungsmöglichkeiten. Dabei wird der Aufbau Ihres 128ers prima erklärt: Betriebssystem, Interpreter, Zeropage, Pointer und Stacks sind nur einige Stichworte dazu. Der erste Schritt hin zur Maschinensprache!

Liesert/Linden
PEEKS & POKES zum C-128
ca. 250 Seiten, DM 29,–
ISBN 3-89011-138-6
Erscheint Ende November



Lassen Sie sich verzaubern! Durch die Grafikmöglichkeiten des C-128. Das BASIC 7.0 enthält viele Befehle, die das Programmieren der Grafik erheblich vereinfachen. DATA BECKER Grafikspezialisten zeigen Ihnen, wie man diese Befehle benutzt, z.B. für Hi-Res/Multi- und Extended Color Grafiken, für Sprites, 3-D, Soft Scrolling und Konstruktionsprogramme. Selbstverständlich wird auch der VIC-Chip mit seinen Registern genau erklärt!

Durben/Löffelmann/Plenge/Vüllers
Das Grafikbuch zum C-128
ca. 300 Seiten, DM 39,–
ISBN 3-89011-154-8
Erscheint im November

DAS STEHT DRIN:

Das große Floppybuch zur 1570/1571 gibt Ihnen das notwendige Wissen zur Programmierung Ihres neuen Diskettenlaufwerkes. Für Anfänger, Fortgeschrittene und Profis. Dieses Buch beschreibt wirklich alle Leistungsmerkmale dieser schnellen Floppy.

Aus dem Inhalt:

- Einführung für Einsteiger
- Die Floppy und das COMMODORE-BASIC
- Sequentielle und relative Dateien
- Fremde Diskettenformate verarbeiten
- Programmierung im DOS-Puffer
- Die CP/M-Fähigkeiten der 1570/71
- Floppy intern: Schaltungsaufbau und Funktion
- 1571 Fast-Load
- Das DOS im Detail
- Komplettes DOS-Listing (mit Cross-Reference)

UND GESCHRIEBEN HAT DIESES BUCH:

Rainer Ellinger ist langjähriger Mikrocomputer-Spezialist mit sehr weitreichenden Assembler- und Hardwarekenntnissen. Sein besonderes Anliegen war es, die teils komplizierten Sachverhalte in eine auch für den weniger versierten Leser, verständliche Sprache zu verpacken.

ISBN 3-89011-124-6